(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: 0 522 780 A2

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 92306064.4

(22) Date of filing: 01.07.92

(51) Int. Cl.^{5}: G06F 3/06

(30) Priority: 12.07.91 JP 197318/91

(43) Date of publication of application: 13.01.93 Bulletin 93/02

(84) Designated Contracting States: DE FR GB

(71) Applicant: International Business Machines Corporation
Old Orchard Road
Armonk, N.Y. 10504 (US)

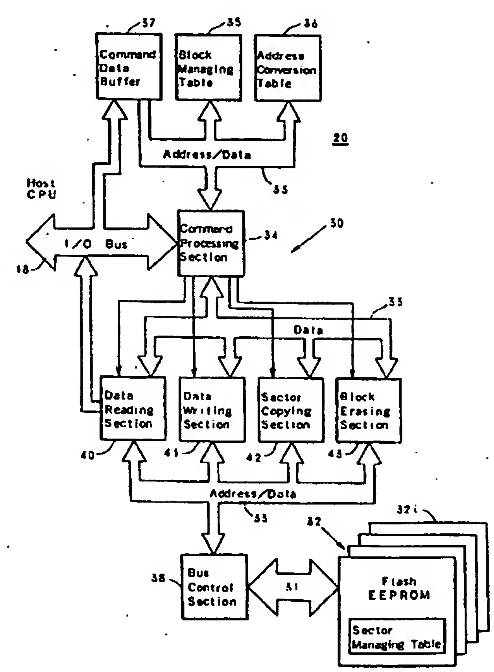
(72) Inventor: Sakaue, Yoshinori
1-8-5-803 Toshima, Kita-ku
Tokyo (JP)
Inventor: Niijima, Hideto
2-42-9-203 Minami-otsuka, Toshima-ku
Tokyo (JP)

(74) Representative: Blakemore, Frederick Norman
IBM United Kingdom Limited Intellectual Property Department Hursley Park
Winchester Hampshire SO21 2JN (GB)

(54) Control method for a computer memory device.

(57) To provide an external storage system using a semiconductor memory in which the data reading and writing between the host CPU can be processed faster than the conventional magnetic disk, and only a particular sector is not frequently written and erased so that the whole memory is effectively used over a long period of time, an address control scheme is introduced in which flexibility is given to the address relation between the host CPU and the external storage and the physical address of the semiconductor memory is not one-sidedly determined by the logical address possessed by the command of the host CPU.

FIG. 2



EP 0 522 780 A2

The present invention relates to computer memory devices using electrically erasable nonvolatile memory, known as flash EEPROM or flash memory and, more particularly, to a method for controlling such a memory for use in e.g. semiconductor external storage systems which can replace the magnetic disk in a conventional computer system.

As portable personal computers such as notebook type ones have spread, the requirement for small-sized, lightweight and low power consumption computer systems has increased. External storage using semiconductor memory has a low power consumption and can operate at a high speed because it does not use a mechanical drive system such as the magnetic disk, in addition, it consists of small memory modules, and thus it is small-sized, lightweight, and has a large degree of freedom with respect to shape as compared with the magnetic disk, and is also easily made in the form of a card.

However, the conventional semiconductor memory has many problems with respect to such points as capacity and battery backup. If SRAM is used as the memory, the cost is high and the capacity becomes small though the backup time by a battery becomes long. For DRAM which is excellent in cost and capacity, the standby power consumption is large and the backup time is limited to one week or so. There is also a danger of data loss due to a problem in the battery system. EEPROM is costly though it requires no battery.

A block erase type flash memory has been developed as a memory to solve these problems. Its memory element consists of one transistor as DRAM, and it can be provided with high density and it is expected to have a bit cost equivalent to or less than DRAM (low cost, large capacity) depending on the future market. The memory element is nonvolatile and requires no battery backup. The outline of such flash memory is introduced by Richard D. Pashley et al. in "Flash memories: the best of two worlds", IEEE SPECTRUM, Dec. 1989, pp. 30-33. Also, a similar flash memory is disclosed in Japanese Published Unexamined Patent Application No. 2-10598.

However, the flash memory has limitations which SRAM and DRAM do not have. First, the erase/program cycles have an upper limit of the order of 100,000. In addition, the programming of memory bits is a one-way process and change is allowed only from 0 to 1 or 1 to 0. For change in the opposite direction, it is necessary to set the whole memory block to 0 or 1 by block erasure. Erasing usually takes several tens of milliseconds, and block erasure requires special procedures such as verify which takes a further several seconds.

If a semiconductor memory consisting of such flash memory is connected to the bus of a host computer as an alternative to the traditional magnetic disk, a specific sector is very often written by the host computer and reaches the upper limit of the erase/program cycles considerably earlier than other sectors. Further, the sector writing takes a lot of time. The reason for this is that, to modify several bytes in the sector, all the data within the memory block including the sector are temporarily saved in the memory space of the host computer, and new data is written back to empty sectors after erasing of the memory block. It takes several seconds to write a sector in a semiconductor memory having a capacity of 1M bits or more. In addition, a special program is required to connect a semiconductor memory to the bus of the host computer.

To solve these problems, the development of a sector erase type flash memory is needed. For instance, 27F010 from SEEQ TECHNOLOGY CORPORATION (1024K flash EPROM) allows not only the chip erasure which erases all the bits of the memory chip (change to logical one) but also the sector erasure which erases only a specific sector. In this scheme, the saving or writing back of sectors other than those to be erased can be avoided. However, it takes on the order of several tens of milliseconds because the erasing of old sectors is done along with the writing of sectors, and thus the obtained performance is equivalent to or less than the magnetic disk. In addition, the problem is not solved that a particular sector is written very often by the host computer and reaches the allowed erase/program cycles somewhat earlier than other sectors, and thus the sectors cannot effectively be used as the whole flash memory. Further, the sector erase type has a chip structure which is more complicated than the block erase type, and that there are also problems with respect to cost and erase/program cycles.

Many of these problems are considered to be due to the control scheme in which the block and sector addresses of the external storage are controlled by the host computer, that is, the physical address of the external storage is determined by the logical address possessed by the command of the host computer.

This invention provides a method for controlling a storage system comprising a semiconductor memory of a flash memory type, the memory comprising a plurality of memory blocks each including at least one sector, and a control device for controlling said memory blocks, said storage system being connected to a host processor through a bus for transferring data and a command; the method comprising selecting a memory block for which data is to be written or erased on the basis of a memory managing table which has recorded therein the number of times each memory block was erased and the statuses of each memory block and each sector; converting the logical address included in said command of said host processor to a physical address indicating the sector in a particular memory block by reference to an address conversion table; responding to said command to perform a proc-

ess of data writing or reading to the corresponding sector or memory block erasure, sequentially recording or updating in said memory managing table the status changes of each sector and memory block resulting from the processes in said block managing means, and storing or updating in the address conversion table the relationships between the physical address and the logical addresses of the memory block and sector on which said processes were performed.

Advantageous features of embodiments of the invention are set out in the attached dependent claims. The present invention enables the provision of an external storage system using a semiconductor memory which can process the data reading and writing between the host computer at a speed higher than the conventional magnetic disk. The whole semiconductor memory is effectively used for a long period of time without frequently writing and erasing only a particular sector in the semiconductor memory. The semiconductor external storage system is compatible with a conventional magnetic disk when it is connected to a computer system. Files can easily be restored even if the power is disconnected in a writing.

This is achieved by providing flexibility in the address relation between the host computer and the external storage, and by introducing an address control scheme in which the physical address of the external storage is not one-sidedly decided on by the logical address possessed by the command of the host computer. On the external storage side, a memory block or sector for writing or copying is always prepared for the command processing of the host, and the correspondence relationship of the physical address of the selected memory block or sector with the command of the host computer is recorded and held in an address conversion table. The status of the memory block and sector is recorded and managed in the respective managing tables.

On the external storage side, an optimum memory block or sector can be selected in consideration of the processing speed of the host computer and the utilization efficiency of the semiconductor memory. In addition, the processing speed of the host computer can be increased without waiting for the command of the host, or executing the command processing in parallel with the command of the host. That is, memory blocks in which data can be written or erased are prepared beforehand on the basis of the record in the memory managing means, thereby occasioning a fast processing in response to the command of the host processor. Moreover, since the memory blocks and sectors of the semiconductor memory can be totally managed and the physical address can freely be selected in consideration of the utilization efficiency, the unnecessary frequent erasure of a particular memory block in the semiconductor memory is eliminated and the whole memory is effectively used over a long period of time. Since the host computer can give a command to the external storage without considering the physical address and receive the result of the process, compatibility with the traditional magnetic disk can be maintained. Also, no rewriting of data directly connected with the command address of the host processor is performed, and thus files can easily be recovered.

In accordance with one embodiment of the present invention, a semiconductor external storage system comprises: a semiconductor memory consisting of a flash memory and comprised of a plurality of memory blocks each including at least one sector, an address conversion table, block managing means, and a command processing section; the address conversion table is used to record the correspondence relationship of the physical address of a sector or memory block with the command of the host processor, and the block mananging means records the number of times each memory block was erased and the status of each memory block and sector; the command processing section selects a memory block in which data writing or erasing is performed on the basis of the record of the block managing means, executes a process of data writing, reading or erasing to the memory block or sector that has the physical address obtained from the address conversion table in response to the command, sequentially records and updates the status change of each sector or memory block in the block managing means, records and updates the correspondence relationship of the physical address of the sector or memory block with the logical address from the host processor in the address conversion table, and selects a memory block in which a data writing or erasing is to be executed next on the basis of the record of the block managing means.

Embodiments of the invention will now be described, by way of example only, with reference to the accompanying drawing, wherein:

Figure 1 shows an example of the computer system in which the semiconductor external storage system of the present invention is incorporated;

Figure 2 shows the detail of the functions of an embodiment of the semiconductor external storage system of the present invention;

Figure 3 shows the details of the construction used for implementing the functions of the semiconductor external storage system of Figure 2;

Figure 4 shows an example of the sector construction of a memory block;

Figure 5 shows an example of the construction of the address conversion table;

Figure 6 shows showing an example of the construction of the block managing table;

Figure 7A is a flowchart showing the normal processing of the command processing section;

Figure 7B is a flowchart showing the normal processing of the command processing section;

Figure 7C is a flowchart showing the normal

processing of the command processing section;

Figure 8 is a flowchart showing the interruption processing of the command processing section;

Figure 9A is a flowchart showing the process of initializing the managing table and conversion table in Figure 7A;

Figure 9B is a flowchart showing the process of initializing the managing table and conversion table in Figure 7A;

Figure 10 is a flowchart showing the detail of the file recovery processing in Figure 7A;

Figure 11 is a flowchart showing the detail of the sector read processing in Figure 8;

Figure 12 is a representation explaining the sector write processing;

Figure 13A is a flowchart showing the detail of the sector write processing in Figure 7A;

Figure 13B is a flowchart showing the detail of the sector write processing in Figure 7A;

Figure 13C is a flowchart showing the detail of the sector write processing in Figure 7A;

Figure 14 is a representation explaining the sector erase processing in Figure 7A;

Figure 15 is a flowchart showing the detail of the sector erase processing in Figure 7A;

Figure 16 is a flowchart showing the detail of the compare processing of i and ie in Figure 15;

Figure 17 is a flow chart showing the detail of the block erase determination and start in Figure 15;

Figure 18 shows an explanatory view of an example of sector erasing by comparison of the file location information;

Figure 19 is a flowchart showing the details of the sector erasing of Figure 18;

Figure 20 is a time chart diagram showing the relationship between the normal processing of the command processing section in Figure 2 and the interruption processing of CPU;

Figure 21 shows an example of the computer system in which the semiconductor external storage system according to another embodiment of the present invention is incorporated;

Figure 22 shows an example of the computer system in which the external storage system according to still another embodiment of the present invention is shown;

Figure 23 is a flowchart showing part of the normal processing of the command processing section in Figure 22;

Figure 24 is a flowchart showing the processings of the sector erase determination and start in Figure 22;

Figure 25 is a time chart diagram showing the relationship between the normal processing of the command processing section and the interruption processing of the CPU in Figure 22.

Figure 1 shows an example of the computer system in which the semiconductor external storage system of the present invention is incorporated; CPU 10 communicates via system bus 13 with main memory 15, bus control unit 16 and coprocessor 14 for arithmetic operations which is optional. The communications between CPU 10 and its associated peripheral devices are done via bus control unit 16. For this, bus control unit 16 is connected to the peripheral devices via family bus 18. As the peripheral devices, external storage system 20 made of a flash memory which is an embodiment of the present invention is connected, and communication device 21, hard file 22 and optical disk 23 are also connected to family bus 18. 24 is a display controller and 25 is a CRT. Of course other peripheral devices can be connected. Such a computer system can be built from, for instance, an IBM PS/2 personal computer (IBM and PS/2 are trademarks of International Business Machines Corporation).

Direct memory access control unit (DMAC) 12 is provided to enable memory accesses by all or selected peripheral devices. For this, at least part of family bus 18 is multipoint-connected to DMAC 12. Although not shown in the figure, an arbitration circuit is provided in each peripheral device to which the DMA access is allowed, and assigned an arbitration level (priority). On the DMAC 12 side, central arbitration control circuit 11 is provided which performs arbitration work between a plurality of peripheral devices demanding a DMA access at the same time and notifies DMAC 12 of which peripheral device has been allowed a DMA access. For details of the DMA control by DMAC 12 and central arbitration control circuit 11, reference is made to the US Patent No. 4,901,234.

Figure 2 shows details of the functional structure of semiconductor external storage system 20. This semiconductor external storage system includes control unit 30 connected to family bus 18, and flash memory 32 connected to semiconductor memory bus 31. Control unit 30 includes command processing section 34, block managing table 35, address conversion table 36 and buffer 37 which are interconnected by control unit bus 33. Flash memory 32 consists of a plurality of memory blocks 320 (321-32i) each containing a plurality of sectors. In this embodiment, as memory block 320, a flash memory chip is assumed which allows only a change from 1 to 0 and is all set to 1 by block erasure, but of course it should be noted that a chip could be used which is opposite to those. Command processing section 34 includes an arbitration circuit and performs a DMA transfer and I/O control. Address conversion table 36 is to convert the logical address sent form CPU 10 through family bus 18 to the physical address of flash memory 32, and consists of a random access memory. This random access memory can include buffer 37.

Bus control section 38 has a well-known receiver/driver used for interconnecting semiconductor memory bus 31 and control unit bus 33. Although bus control section 38 is constructed so as to accomplish

parallel transfer of 16 bits between buses 31 and 33 in this embodiment, the present invention of course is not limited to such parallel transfer. Command processing section 34 manages data reading section 40, data writing section 41, copying section 42 and block erasing section 43.

Flash memory 32 is managed as a collection of sectors. In this embodiment, individual memory blocks 320 constituting flash memory 32 consist of 512 sectors, and each sector includes 512 bytes. The numbers of sectors and bytes may be changed depending on the storage capacity of memory block 320 and the architecture.

It should be noted that the term memory block as used herein means a region on the memory chip which can be collectively erased by the block erasing section. Although one memory block corresponds to one memory chip in this embodiment, a plurality of memory blocks could exist in one memory chip, and one sector could correspond to one memory block as in the example of the sector erase type memory described later.

Figure 3 shows the hardware arrangement of semiconductor external storage system 20 of Figure 2. In the figure, the respective functional portions of command processing section 34, data reading section 40, data writing section 41, copying section 42 and erasing section 43 are built with a microprocessor 50. The micro code controlling the microprocessor 50 is stored in ROM 52. Since block managing table 35 and address conversion table 36 are often modified and very fast access is required, they are formed on RAM 54. Microprocessor 50 communicates with the host CPU 10 via I/O register 56 at any time. Data transfer control unit 58 connects I/O bus 18 and local bus 33 as needed, thereby enabling data transfer between buffer 37 and flash memory (EEPROM) 32. Except for data transfer, these buses are disconnected and microprocessor 50 can function independently of host CPU 10.

As shown in Figure 4, the first several sectors of each memory block 320 are used as sector managing table 60, in which the status 64 and logical address L 66 of each sector are stored. The remaining sectors of each memory block 320 are used as data region 70 (70A-70N). The size of sector managing table 60 depends on the memory block and the capacity of the sector included therein, and in the above described example of 512 sectors per block and 512 bytes per sector, four sectors are required as the sector managing table. To each sector 70A-70N, a logical address is assigned as described later. These sectors including sector managing table 60 are formatted on a memory chip using a format program.

Sector managing table 60 includes a plurality of entries each of four bytes, and in the first entry thereof, the number of times 62 the related memory block 40 was erased is stored. In the second entry, status 64 and logical address 66 of the first sector (in this case, sector 4) used as data region 70A in the related memory block are stored. In the third entry, status 64 and logical address 66 of the next sector 70B or sector 5 are stored, and similarly, the status and logical addresses of the subsequent respective sectors are sequentially stored.

Status 64 of a sector is indicated by a 4-bit status flag as described below. Since the status flag is on the flash memory, the bit change is limited in one direction.

1111 = blank
1110 = valid
1100 = invalid
0000 = under erasure

The status for each memory block is recorded in block managing table 35 on the basis of status 64 in sector managing table 60. Figure 6 shows an example of block managing table 35, in which the number of blank sectors B (i, 1) 72 and the number of valid sectors B (i, 2) 74 per memory block i, and the number of blank sectors B (, 1) 76 of the whole memory block are recorded.

Returning to Figures 2 and 3, command processing section 34 communicates with CPU 37 via buffer 37. That is, the CPU command and data are DMA-transferred to buffer 37 under control of DMAC 12, and command processing section 34 fetches the command from buffer 37 and executes it. Since flash memory 32 is seen from CPU 10 as if it were a hard disk or floppy disk, the command transferred to buffer 37 is the same type as the command for such disk and contains sector address L in addition to the OP code. In external storage system 20, the sector address L is treated as logical address 66. Address conversion table 36 which provides the relationship between the sector address L (logical address 66) and physical address A (L) (pointer 68) of memory block 40 is held on RAM 54 with the construction as shown in Figure 5.

Now, the operation of control unit 30 of external storage system 20 will be described. Command processing section 34 usually independently executes processes such as sector writing and block erasing. It receives the command and data from host CPU 10 by interruption. The normal processing is outlined in Figures 7 - 8.

First, in the first step of Figure 7 (Figure 7A - 7C), the initialization of the block managing table and address conversion table is done (702). Then, it is determined whether or not file recovery is required for disconnection of the power supply or the like, and the process therefore is executed if necessary (704, 706). Next, it is checked whether there is any command sent from host CPU 10 to buffer 37 (708). If there is a write command, the existence of blank sectors in memory block 40 is checked (total number B (, 1) > 0), and a writing is performed if there is a blank sector (710 - 714). If it is a sector erase command, the

erasing process of the corresponding sector is done (716, 718). If a certain memory block is determined to be a memory block ie to be erased, the block erasing process (later described in detail) is started. In the block erasing process, the sector copy to other memory blocks is undertaken to save valid sectors as described later. When the sector copy is terminated, the command processing section holds the number of times the memory block was erased and thereafter commands block erasing section 43 to execute block erasing of the correspond- ing memory block ie (720 - 724). The number of erasing times is stored and managed in the first four bytes of the sector managing table, and it is incremented by one after completion of the erasure and written back (728). Then, the erased block ie is made to be blank block ib (730), the records of the sector managing table and block managing table 35 are updated. That is, the number B (i, 1) of empty sectors in the memory block i is made to be M and the number B (i, 2) of effective sectors is made to be 0 (732, 734).

However, if the number of erasing times of a certain memory block i to be collectively erased is smaller than that of a memory block ie, an erase candidate, by a certain value, the memory block i is selected as the memory block ie which is the erase candidate used to make uniform the number of erasing times, regardless of whether the number of the "valid" sectors is large or small (736). If there is no further memory block ie to be erased in all the memory blocks and no blank sector iw exists, the request for memory replacement is displayed on CRT 25 (Figure 1) (738, 740). This display may be done earlier so as to leave time to spare.

The request for processing from host CPU 10 to external storage system 20 is accepted by the previously mentioned interruption in the normal processing. As shown in Figure 8, when command processing section 34 accepts a command from host CPU 10 (802), it executes the processing required for a sector reading (804, 806), a sector writing (808, 810) or sector erasing (812, 814). To execute each command, command processing section 34 looks up address conversion table 36 for the logical (sector) address 66 in the command and obtains the corresponding physical address 68. The physical address 68 consists of a blocks address (i) specifying a particular memory block 32i of flash memory 32, and a sector address (j) specifying a particular sector j in that block.

The details of each processing of the command processing section 34 will now be described. First, the initialization of the block managing table and address conversion table (Figure 7A, step 702) is described. Figure 9 (Figures 9A, 9B) shows the details of the initialization process of the block managing table and address conversion table. The block number i and sector number j are initialized, the number of empty sectors B (i, 1) and the number of valide sectors B (i, 2) are both set to zero (902, 904), and the status of the j-th sector in block i and data S (i, j) of pointer L are obtained from sector managing table 60 (906). If the sector is vacant, add one to the number of blank sectors B (i, 1) in block i. If the sector is not vacant, then the sector is checked for validity, and if it is valid, physical address 68 is stored in L-th A (L) of address conversion table 36 (914) and one is added to the number of valid sectors B (i, 2) (916). A similar processing is executed until sector number j reaches the total number of sectors M in block i (920). Further, a similar processing is repeated for all blocks N (922 - 924). The number iw of the block in which a sector writing is performed, the number ie of the candidate block for erasure, and the number ib of the blank block in which data is to be saved for the erasure are determined (926).

The file recovery (Figure 7A, step 706) is described below. In the conventional magnetic disk file, data is overwritten on the sector when the sector is rewritten, and thus, if a power failure occurs in the course of rewriting a file for instance, both old and new files would be lost. On the other hand, in the present invention, a new sector is found for the sector rewriting without overwriting the old sector, and thus the old data continues to be valid until the block including the data is erased (however, the status flag of the sector has already been rewritten to invalid). Accordingly, if a file writing has failed, it is possible in most cases to find old sector 66 from the value of L in S (i, j) and reproduce the file regardless of status flag 64.

Figure 10 shows details of the file recovery. If the power supply is disconnected during a file writing by and accident or the like, the data in buffer 37 or RAM 54 is lost. Then, the old file is recovered when the power is turned on again. First, the directory information and file allocation information are read out to obtain the leading sector number K of the file (1002, 1004). As the sectors corresponding to the sector numbers written until the power was turned off, both invalid sectors having old data and valid sectors having new data exist. Then, an invalid sector having the sector number K is searched, and if there is one present, the data of the sector is moved to a new blank sector after obtaining the user's confirmation to make it valid. The existing valid sector is made invalid (1010). The user's confirmation is obtained because, if the old data comprises plural versions, confirmation is obtained as to which one should be selected. A similar processing is done for all the sectors constituting the file (1014). The old file is reconstructed by gathering the data of invalid sectors in this way.

Figure 11 is a detailed flow of the sector reading (step 806 of Figure 8), in which a physical address A (L) 68 corresponding to a given logical address L 66 is obtained from address conversion table 36, the positions sectors 70A - 70N are found, and data (AL) is set in data reading section 40 and DMA-transferred

to CPU 10.

Now, the sector write processing (step 714 of Figure 7A) is described. In Figures 2 and 12, command processing section 34 first obtains the logical address L given by host CPU 10 from buffer 37, and then looks up address conversion table 36 to obtain physical address A (L). Further, it traverses block managing table 35 to find the position of the blank sector 70N in data region 70. And, it reads out the data received from host CPU 10 from buffer 37, writes this new data into the blank sector 70N, and rewrites its status flag 64N from "blank" to "valid" and the status flag 64A of the old sector 70A from "valid" to "invalid." Then, it stores the correspondence between the logical address 66 and the physical address of the new sector 70N specified by host CPU 10 in address conversion table 36, and the record of the block managing table is renewed as to the number of blank sectors (i, 1), the number of valid sectors (i, 2) and the total number of empty sectors B (, 1). Incidentally, the memory block iw containing the new sector 70N may exist in the same memory block 320 as the old sector 70A or in another memory block.

Figure 13 (13A - 13C) shows details of the above sector write processing.

First, the logical address L included in the command given from host CPU 10 is obtained (1302). And, if the processing of the later described file allocation information is required, that processing is done (1304, 1306) and then the physical address A (L) is obtained from address conversion table 36 (1308). Further, the old memory number i and sector mumber j are obtained for the A (L) (1314). Then, the memory address P of the blank sector in the memory block iw in which the writing is done is obtained, and this P is set in data writing section 41 and the sector data is written (1316 - 1320). In addition, the sector number k of the writing destination is obtained from P, the data S (iw, k) of the pointer is made valid, L is stored, and P is made to be the physical address A (L) (1322 - 1326). Then, in order to logically erase the old data at the same logical address, the value S (i, j) of the old pointer L of the memory block is made invalid (1330) and the number of valid sectors B (i, 2) is decreased by one (1332). Moreover, it is determined whether or not the block i is that to be erased next (1334), and the number of blank sectors B (iw, 1) is reduced by one (1336). And, if the number of blank sectors becomes 0, the memory block having blank sectors is renewed as the memory block iw in which next writing is done (1340). However, the blank memory block selected as the copying memory block for saving the sector data is excluded. Finally, the determination of the block erasing and the processing are done (1342).

The procedure of sector erasing (step 718 of Figure 7B) is now described in detail according to Figures 14 - 15.

In Figure 15, each time the status flag of the sectors of a certain memory block i changes from "valid" to "invalid", command processing section 34 invalidates the values S (i, j) of th pointer (1508) and decrements the number of "valid" sectors B (i, 2) in the memory block i in the block managing table by one (1510). This logically erases the sector data, and physically holds it as it is valid. And, the physical address A (L) of the address conversion table is made to be zero, and the number of the remaining "valid" sectors is compared with the erase candidate memory block ie. If the number is smaller, this memory block i is selected as a candidate ie of the memory block to be erased next (1514).

In this embodiment, the number of "blank" sectors is always kept greater than a fixed value by the block erasing processing based on the block managing table and sector managing table. The "blank" sectors are used as destinations at which data is written or copied at the time of data writing or erasing. As shown in Figure 14, there are many "empty" sectors initially existing in one memory block 321, but "valid" sectors and "invalid" sectors gradually increase. And, when the number of "valid" sectors 70 remaining in the memory block 321 becomes smaller than the predetermined value, the "valid" sectors 70 are copied into "empty" sectors in another memory block 322, and then the content of the memory block 321 is erased in bulk to make them all "blank" sectors.

The command from CPU 10 is executed by interruption even during the processing of copying sectors or erasing memory blocks. For instance, if the command of CPU 10 is for data writing, a memory block 323 other than the memory block 322 which is currently copying data is selected, and a writing process is done in parallel with the copying.
Consequently, the interrupt function (parallel processing) is executed as long as blank sectors are secured, and thus the execution of the command from CPU 10 is not to wait.

Incidentally, since it takes time to copy the data of valid sectors prior to block erasure, the number of copying times is preferably as small as possible. Accordingly, in step 1334 of Figure 13C and step 1514 of Figure 15, a processing is done to minimize the number of valid sectors of the memory block to be erased, thereby suppressing the amount of copying to a small value. For this, if a certain sector now becomes invalid, the number of valid sectors of the memory block i including that sector is compared with the number of valid sectors of the erasing candidate memory block ie to determine the next erasing candidate. The details of this are shown in Figure 16. First, the number of erasing times E (i) of a memory block i in which a sector has now become invalid is compared with the upper allowance value X of the number of erasing times of that memory. This upper limit value X is set to, for instance, 10,000 times or 100,000 times depending on the specification of the

memory block. If X is exceeded or there are blank sectors B (i, j) in the memory block i, it is not necessary to change the erasing candidate and the determination terminates (1602, 1610). If E (i) has not yet reached the upper limit X, then it is checked whether or not the difference between the memory block having the largest number of erasing times Q among the all memory blocks and the number of erasing times E (ie) of the erasing candidate memory block ie is equal to or smaller than the maximum allowed value R for deviation of the number of erasing times (1604). R, an appropriate value between 100-1000 is selected for instance. If the deviation is larger than R, the examination of the memory block i is not required and the determination terminates. If, conversely, it is smaller than R, comparison of the number of valid sectors B (i, 2) is done (1606). If the memory block i has only valid sectors B (i,2) less than the erasing candidate memory block ie, the memory block i is newly replaced as the next erasing candidate memory block ie (1608). Incidentally, since the total number of sectors in the memory block i is known, it will be understood that the comparison of i and ie may be done by the number of invalid sectors instead of the number of valid sectors. Referring to Figure 17, the detail of the determination processing of block erasing (step 1342 of Figure 13 C and step 1516 of Figure 15) is described. First, the number of valid sectors B (ie, 2) of the erasing candidate memory block ie is compared with a predetermined value S (1702), and if the number of valid sectors becomes smaller, sector copying section 42 is instructed to copy and save the valid sectors in the block ie to the memory block ib for copying (1706). The next writing or copying processing is hindered if the number of the blank sectors in all the memory blocks is small though there are many valid sectors. Thus, if the total number of blank sectors B(, 1) is smaller than a predetermined value T, sector copying processing is similarly performed for erasing (1704, 1706).

To efficiently reproduce blank sectors, it is needed to know the sectors erased by the host CPU as early as possible. However, the conventional file system only updates the file allocation information when a file is erased by CPU 10, and it does not inform the external storage of which sectors were erased. In accordance with the embodiment of the present invention, when the file system was rewritten (step 1304 of Figure 13A), this problem can be solved by comparing the file allocation information as shown in Figs. 18 and 19 without rewriting the file system. That is, command processing section 34 has a function which compares the new and old file allocation information FAT. For this, the file allocation information is utilized for which CPU 10 manages the logical address L. When the file system is rewritten by CPU 10, that information is stored at a specific logical address L by the operating system OS. Then, the old file allocation information FATO on the external storage side is read out (1902) and compared with each entry U of the new file allocation information FATN at the CPU 10 side (1908), and if there is a sector 70N which was newly freed, sector erasing processing of the sector 70N is done (1912). A similar processing is performed for all entries to update the sector status on the external storage side.

In accordance with the above described embodiment of the present invention, all the sectors are substantially uniformly used and a high-speed sector writing is enabled. Figure 20 shows a comparison of the processing times for sector writing in the conventional scheme (A) and the scheme of the present invention (B). In the conventional scheme, a block is erased after all the data in the block are saved in the main memory or buffer at the beginning, and thereafter the data are written back to the memory block along with the newly updated data. Usually there are about 64-256 sectors in a block, and it takes a considerable amount of time to read out and write back all of them. The processing in the host CPU is interrupted during that time. On the other hand, in the scheme of the present invention (B), as the processing by CPU 10 for sector writing, it is only needed to write one sector, which eliminates the need for the data saving processing that has conventionally been required, and the block erasing is independently executed in command processing section 34 in parallel with the (other) processing of CPU 10. Accordingly, a fairly high speed sector writing as compared with the conventional scheme, in other words, the shortening of the binding time of CPU 10 is allowed. The reason for this is that the present invention manages memory blocks and sectors so as to always separately prepare a memory block in which sectors are written a memory block to be erased and a blank memory block. This allows the parallel execution of a sector writing and a memory block erasing, and a sector writing and a memory block erasing.

Figure 21 shows another embodiment of the present invention, which is different in that command processing function 34 is provided by host CPU 10, as compared with the example of Figure 2. Also, block managing table 35 and address conversion table 36 are located on main memory 15 (including a buffer) and referenced by the host CPU. Host CPU 10 sends an instruction to memory controller 80 via I/O bus 18 for a sector reading or writing. The memory controller is formed as an integrated circuit on a silicon substrate which is separate from the host CPU, and provides the functions of data reading 40, data writing 41, sector copying 42 and block erasing in addition to buffer 37. Memory controller 80 can access to flash memory 32 independently of host CPU 10, and notify the host CPU of the end of a task via I/O bus 18. Since the other operations are the same as the embodiment of Figure 2, the explanation thereof is omitted.

Further, Figure 22 shows another embodiment of the present invention which uses a sector erase type flash memory as memory block 32. The sector erase type is a type in which the size of a memory block erased in bulk is equal to the physical size of the file sector. Since the erasing is performed for each sector in this example, the sector saving prior to block erasure is not necessary as compared with the example of Figure 2, and accordingly there is no sector copying section 42, and instead of block erasing section 43, sector erasing section 82 is formed on a microprocessor to perform the saving and recovery of the number of erasing times of sectors. In addition, sector managing table 60 is provided for each unit of memory block 320, or sector 70. The status and the number of easing times of sectors are recorded in sector managing table 60, and empty sectors for writing are secured in consideration of the number of erasing times of all the sectors. The normal processing of the command processing section is used to check the buffer, and if there is a command, executes a sector writing or erasing as shown in Figure 7. However, as shown in Figure 23, the processing performing a sector copying prior to a sector erasing is unnecessary, and the erasing of invalid sectors is immediately executed (727-733). Further, as shown in Figure 24, processing is required to save the number of erasing times of a sector prior to the erasing of the sector and for writing it back (1705-1707).

Also in this embodiment, as shown in Figure 25, all sectors are almost evenly used and high-speed sector writing is enabled. In this example, since the erasing is done on a sector basis, the saving of data and the copying of sectors are not necessary. In the conventional scheme (A), the host CPU also needs to do the erasing of a sector each time the sector is written. In the scheme of the present invention (B), the processing of the host CPU is only used to write a sector and the other processings, for instance, sector erasing is independently processed by the memory controller, and thus the processing time of the host CPU may be short and a high-speed sector writing is made possible.

Incidentally, if the semiconductor has a large capacity, the semiconductor memory may be divided into a plurality of groups each having a plurality of memory blocks, and the above described managing and control of blocks and sectors may be performed for each group.

In accordance with the present invention, a semiconductor external storage system using a flash memory is used which can process data reading and writing at high speed in response to the command of the host CPU. In addition, an external storage system is obtained in which, even if the command of the host CPU frequently rewrites a particular sector of the flash memory, the number of erasing times of sectors are not biased on the actual flash memory and the whole can be effectively utilized.

In a computer system comprising a host processor, a semiconductor memory is used as an external storage device consisting of a flash memory and comprised of a plurality of memory blocks each including at least one sector, and a control section performing control over said memory blocks, a method has been described for controlling the external storage device of a computer system wherein said control section converts the logical address included in the command of said host processor to a physical address indicating the sector in a particular memory block by address conversion means, selects a memory block for data writing and a memory block for erasure to be erased next to acquire said memory block for data writing, respectively, on the basis of the record of block managing means which has sequentially recorded therein the number of times each memory block was erased and the status of each sector and memory block, writes the data to be written received from said host processor into said memory block for writing, and erases said selected memory block for erasure.

To provide an external storage system using a semiconductor memory in which the data reading and writing between the host CPU can be processed faster than the conventional magnetic disk, and only a particular sector is not frequently written and erased so that the whole memory is effectively used over a long period of time. An address control scheme was introduced in which flexibility is given to the address relation between the host CPU and the external storage and the physical address of the semiconductor memory is not one-sidedly determined by the logical address possessed by the command of the host CPU. Command processing section 34 always prepares memory blocks and sectors for writing or erasing and copying in preparation for the command processing of the host CPU, and records and holds the correspondence relation between the physical address of the selected memory block 32i or sector and the command of the host CPU in address conversion table 36. The status of memory blocks and sectors is recorded in respective managing tables 35 and 60, and used for control of processings such as writing, erasing and copying in preparation for or in response to the command of the host CPU.

Claims

1. A method for controlling a storage system comprising a semiconductor memory of the flash memory type, the memory comprising a plurality of memory blocks each including at least one sector, and a control device for controlling said memory blocks, said storage system being connected to a host processor through a bus for transferring

data and a command, the metod comprising;

selecting a memory block for which data is to be written or erased on the basis of a memory managing table which has recorded therein the number of times each memory block was erased and the statuses of each memory block and each sector;

converting the logical address included in said command of said host processor to a physical address indicating the sector in a particular memory block by reference to an address conversion table;

responding to said command to perform a process of data writing or reading to the corresponding sector or memory block erasure;

sequentially recording or updating in said memory managing table the status changes of each sector and memory block resulting from the processes; and

storing or updating in the address conversion table the relationships between the physical addresses and the logical addresses of the memory block and sector on which said processes were performed.

2. A method as claimed in Claim 1 comprising accepting the command of said host processor by interruption, and if said command is a data read command, responding thereto to perform the process of reading the data from the corresponding sector of said memory block corresponding to the logical address of said command, and if said command is a write or erase command, holding such command and data in a buffer and executing said write or erase process to said selected memory block or sector when there is no interrupt command from said host processor.

3. A method as claimed in Claim 1 or Claim 2, comprising selecting a particular memory block of said memory blocks with priority as an object to be erased if the number of times said particular memory block was erased is smaller than the largest number of times any other memory block was erased by a predetermined value.

4. A method as claimed in any of Claims 1 to 3 comprising selecting based on the record of said block control means, a memory block having free sectors for copying the data in the valid sectors of said memory block to be erased, and

copying said data into said selected memory block prior to the erasure of said memory block.

5. A method as claimed in any preceding claim comprising executing the process of reading the data from a corresponding sector of said memory block and said write or erase process to said memory block or sector by a time sharing process.

6. A method as claimed in Claim 5 comprising responding to a sector write command from said host processor by writing said data into a free sector of said selected memory block, and if the logical address of said sector write command is equal to the logical address of the valid sector which has already been written, rewriting the record of the status with respect to said valid sector of the memory management table to be invalid, and writing the physical address of said free sector into said address translation table.

7. A method as claimed in any preceding claim comprising saving the record of the number of times a memory block has been erased prior to each erasure of said memory block, and writing said record back to said memory block after erasure of said memory block.

8. A method as claimed in any preceding claim comprising updating the record of said memory management table on the erasure of a memory block by comparison of the new file allocation information held by said host processor with the old file allocation information held in the external storage system.

9. An external storage system connected to a host processor through a host bus for transferring data and a command comprising:

a semiconductor memory consisting of a flash memory and comprised of a plurality of memory blocks each including at least one sector,

address conversion means for converting the logical address received from said host processor to the physical address of a particular sector,

block managing means for recording the status of each memory block and sector, and

a command processing section which uses the record of said block managing means to control the reading or writing of data to each sector of said semiconductor memory, or the erasing of a memory block.

10. An external storage system as claimed in Claim 9 wherein said control section includes a data writing section, data reading section and block erasing section for executing a data writing, reading or erasing process to the corresponding memory block or sector of said semiconductor memory on the basis of said command processing section, and a sector copying section for copy-

ing the valid sectors of the memory block to be erased to another memory block.

11. An external storage system as claimed in Claim 9 wherein said control section includes:

a command processing section, data writing section, data reading section, block erasing section, and a sector copying section used for copying the valid sectors of a memory block to be erased to another memory block which consists of a microprocessor, a block managing table, address conversion table and a buffer which are formed on RAM, and

a sector managing table constituting said block managing means along with said block managing table provided on each memory block.

12. An external storage system as claimed in Claim 9 wherein said control section includes:

a command processing section formed on a host processor,

a block managing table and an address conversion table formed on the RAM of the main memory,

a memory controller including a buffer, and

a sector managing table constituting said block managing means along with said block managing table provided on each memory block.

13. A computer system comprising a host processor, an external storage system including a semiconductor memory which consists of a flash memory and comprises a plurality of memory blocks each including at least one sector, and a host bus for transferring data and a command between said external storage system and said host processor, characterized by:

control means which uses an address conversion means for converting the address from said host processor to the physical address of a particular sector and block managing means for recording the status of each memory block and each sector therein to control the data reading and writing to each sector of said semiconductor memory and the erasure of a memory block in response to said command,

said control means having means which are responsive to the command from said host processor for executing the process of data writing or reading to the corresponding sector of said semiconductor memory or of erasing a memory block, and for updating the record of the status of each memory block and each sector of said block managing means in connection with such process.

14. A computer system as claimed in Claim 13 having display means connected to the host processor and the external storage system through the host bus, wherein

said control means outputs to said display means that said semiconductor memory is to be replaced when there is no free sector in any of said memory blocks and there is also no memory block to be erased.

15. A computer system as claimed in claim 13 having interactive display means connected to the host processor and the external storage system through the host bus, wherein

said control means receives information as to whether or not file recovery is required from a user by an interaction via said display means, and if file recovery is required, refers to the file allocation information held by the computer system and displays the content of the file, and obtains the user's confirmation, thereby performing the file recovery.

16. A semiconductor memory consisting of a flash memory and comprising a plurality of memory blocks each including at least one sector, wherein

a particular region is provided in each said memory block for storing the record representing the number of times said memory block was erased and status of each sector.

17. A semiconductor memory as claimed in Claim 16 wherein one said particular region is provided for each memory block as a sector managing table for recording the status of all the sectors included in each memory block.

F I G . 1

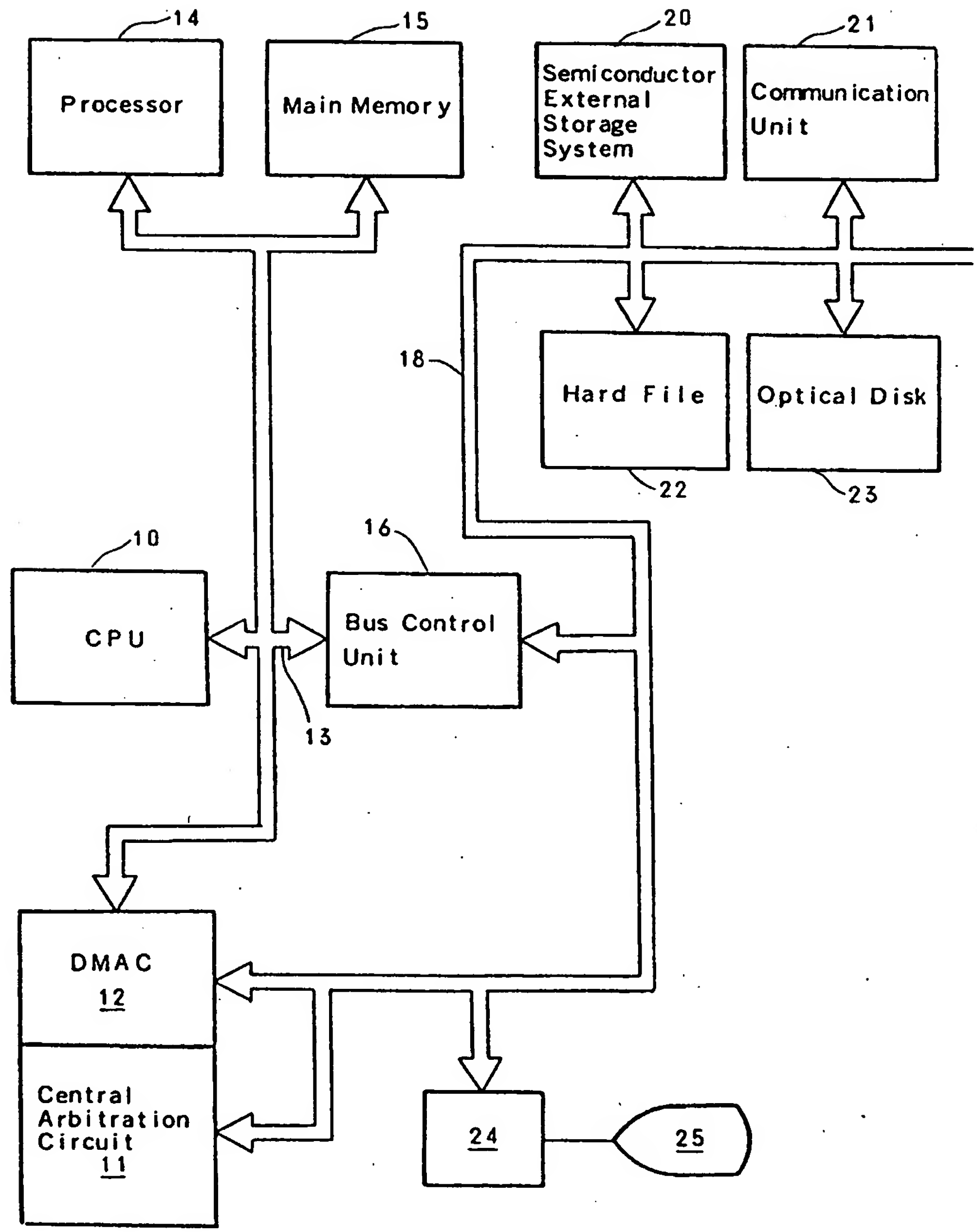

14
Processor
15
Main Memory
20
Semiconductor External Storage System
21
Communication Unit
18
Hard File
22
Optical Disk
23
10
CPU
16
Bus Control Unit
13
DMAC
12
Central Arbitration Circuit
11
24
25

FIG. 2

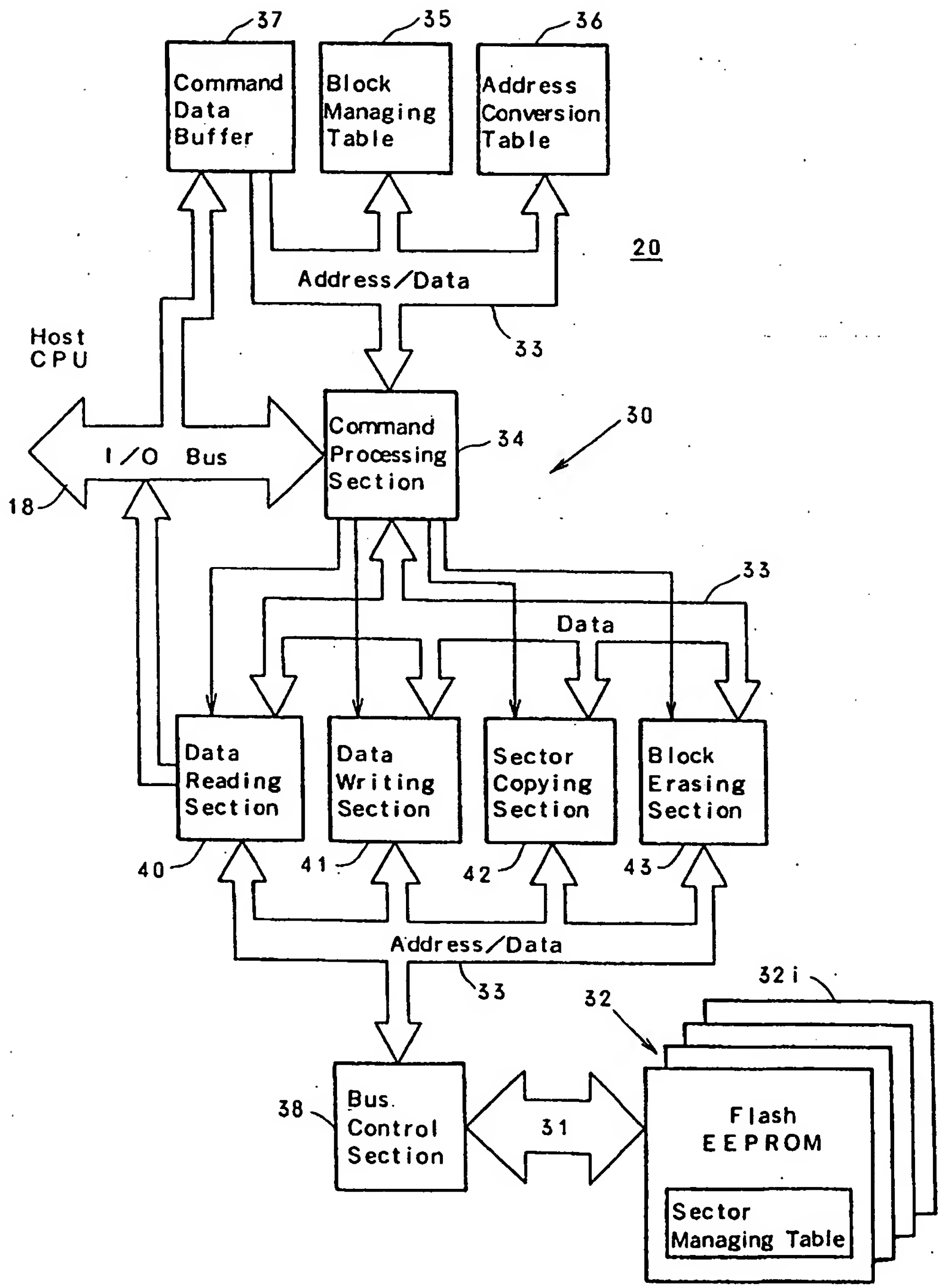

37
Command Data Buffer
35
Block Managing Table
36
Address Conversion Table
20
Address/Data
33
Host CPU
I/O Bus
18
Command Processing Section
34
30
33
Data
Data Reading Section
Data Writing Section
Sector Copying Section
Block Erasing Section
40
41
42
43
Address/Data
33
32i
32
38
Bus Control Section
31
Flash EEPROM
Sector Managing Table

FIG. 3

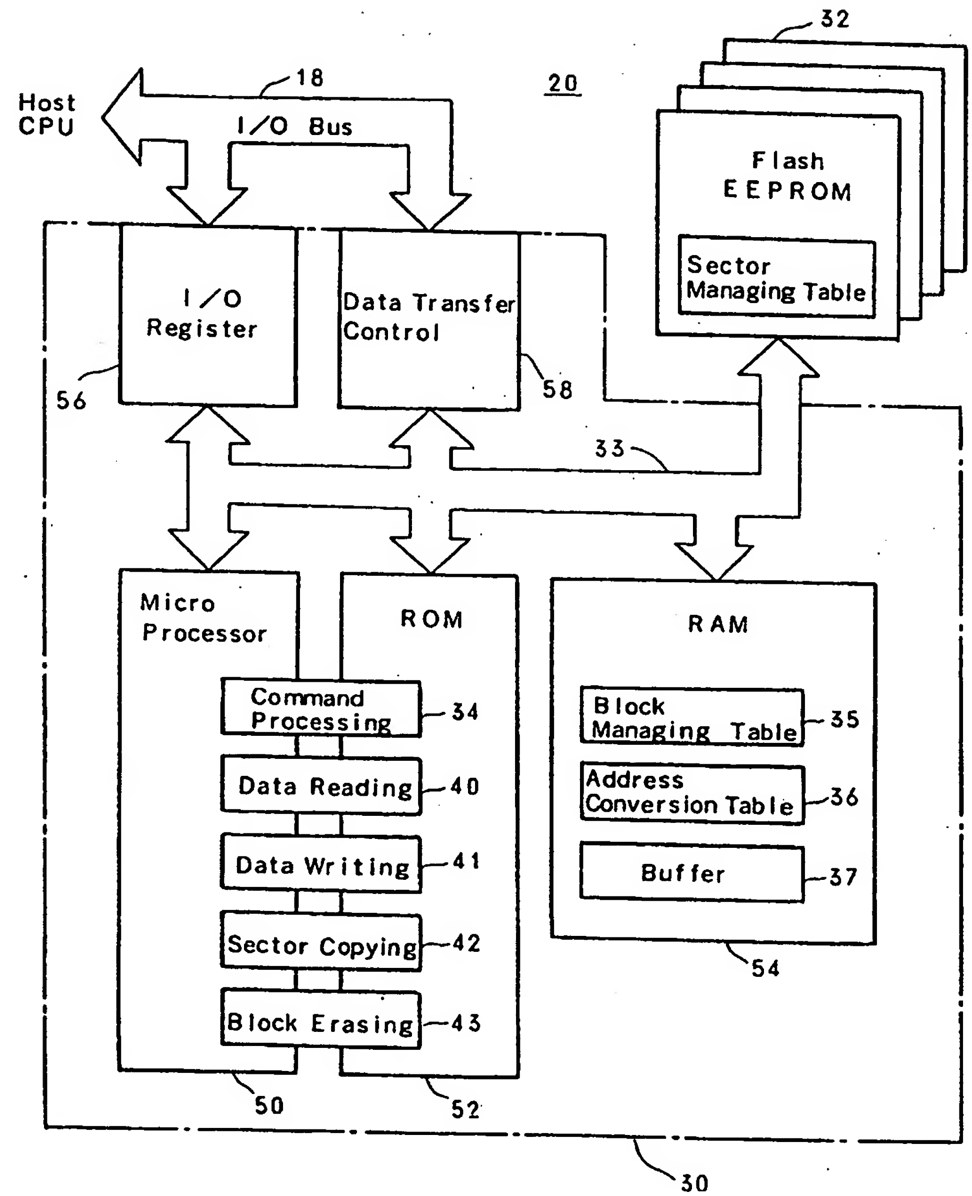
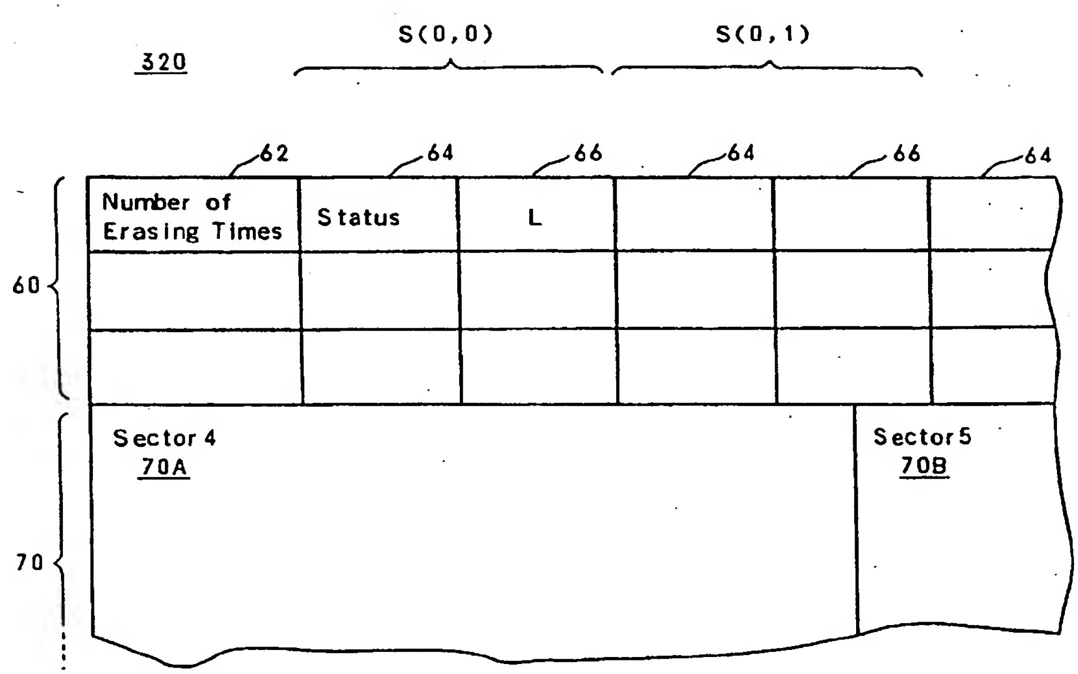

Host CPU
18
I/O Bus
20
32
Flash EEPROM
Sector Managing Table
I/O Register
Data Transfer Control
56
58
33
Micro Processor
ROM
RAM
Command Processing
34
Data Reading
40
Data Writing
41
Sector Copying
42
Block Erasing
43
Block Managing Table
35
Address Conversion Table
36
Buffer
37
54
50
52
30

FIG. 4



320
S(0,0)
S(0,1)
62
64
66
64
66
64
Number of Erasing Times
Status
L
60
Sector 4
70A
Sector 5
70B
70

FIG.5

36

66 68

L	A(L)
1	1st pointer
2	2nd pointer
3	3rd pointer
⋮	⋮
N	N-th pointer

FIG.6

35

72 74

i	Number of Empty Sectors	Number of Valid Sectors
0	B(0,1)	B(0,2)
1	B(1,1)	B(1,2)
2	B(2,1)	B(2,2)
⋮	⋮	⋮
N-1	B(N-1,1)	B(N-1,2)
	B(,1)	

76

FIG.7A

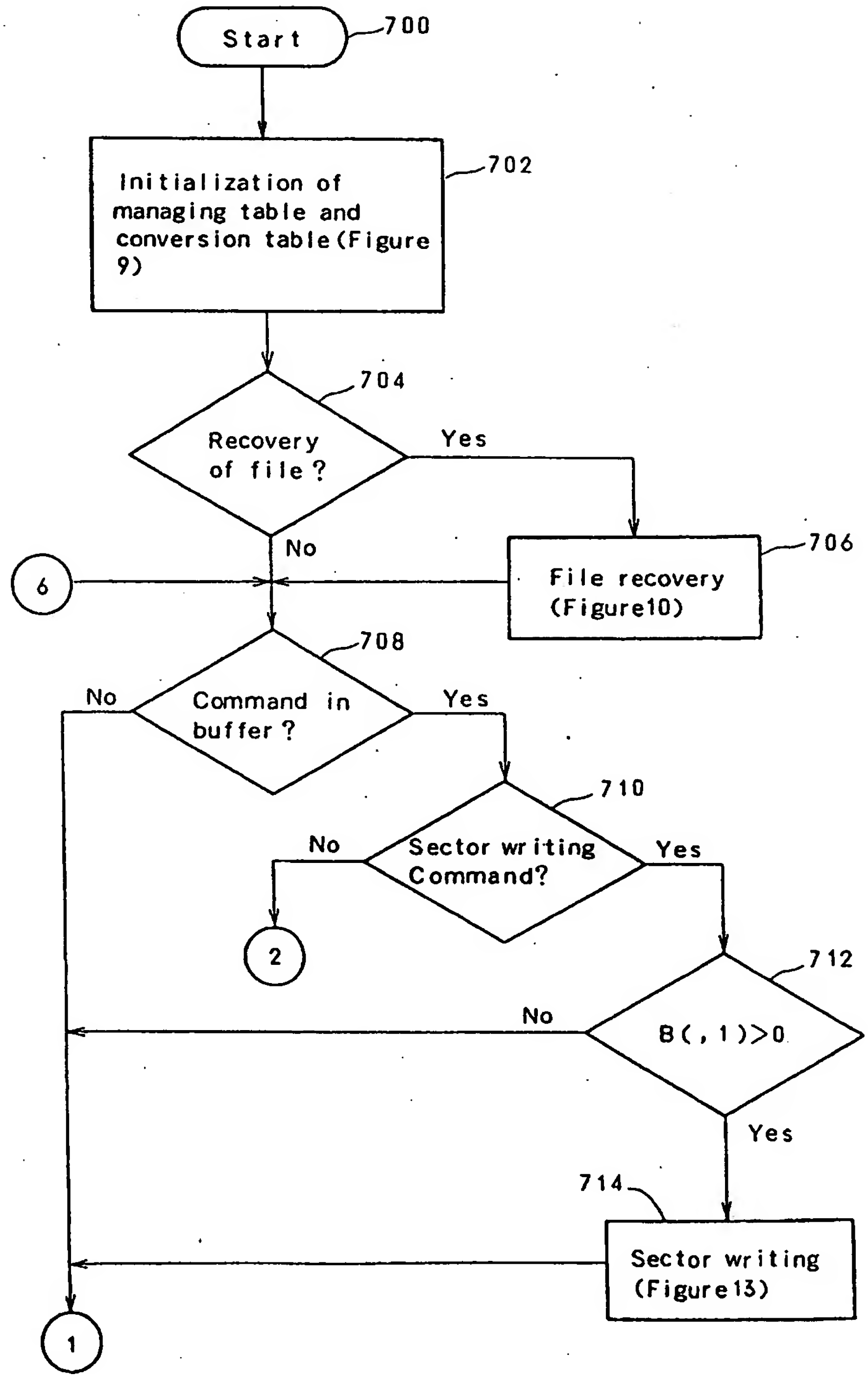

Start
700
Initialization of managing table and conversion table (Figure 9)
702
Recovery of file ?
704
Yes
No
File recovery (Figure10)
706
6
Command in buffer ?
708
No
Yes
Sector writing Command?
710
No
Yes
2
B(, 1)>0
712
No
Yes
714
Sector writing (Figure 13)
1

FIG.7B

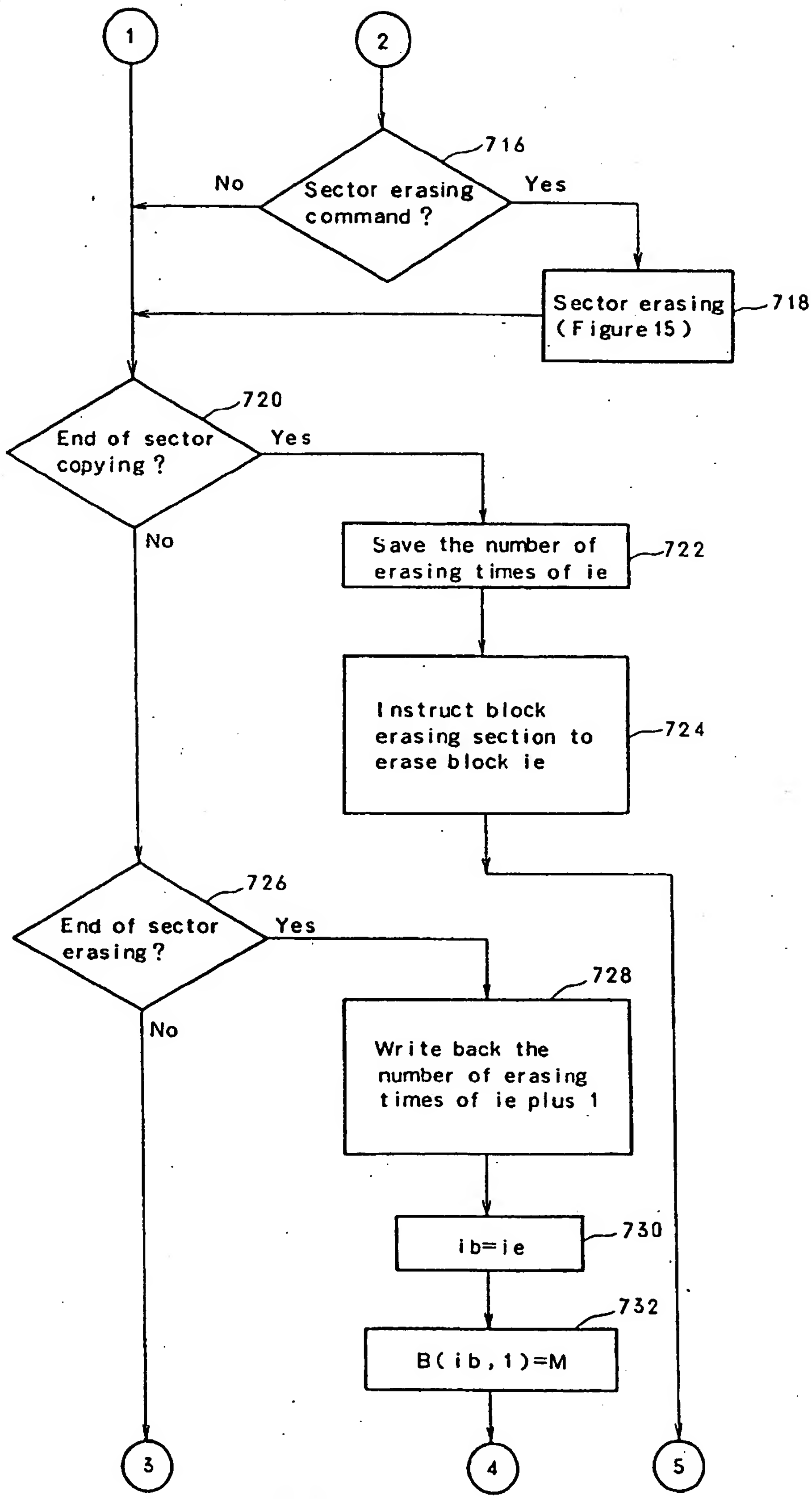

1
2
716
Sector erasing command ?
No
Yes
Sector erasing (Figure 15)
718
720
End of sector copying ?
Yes
No
Save the number of erasing times of ie
722
Instruct block erasing section to erase block ie
724
726
End of sector erasing ?
Yes
No
728
Write back the number of erasing times of ie plus 1
ib=ie
730
732
B(ib,1)=M
3
4
5

FIG.7C

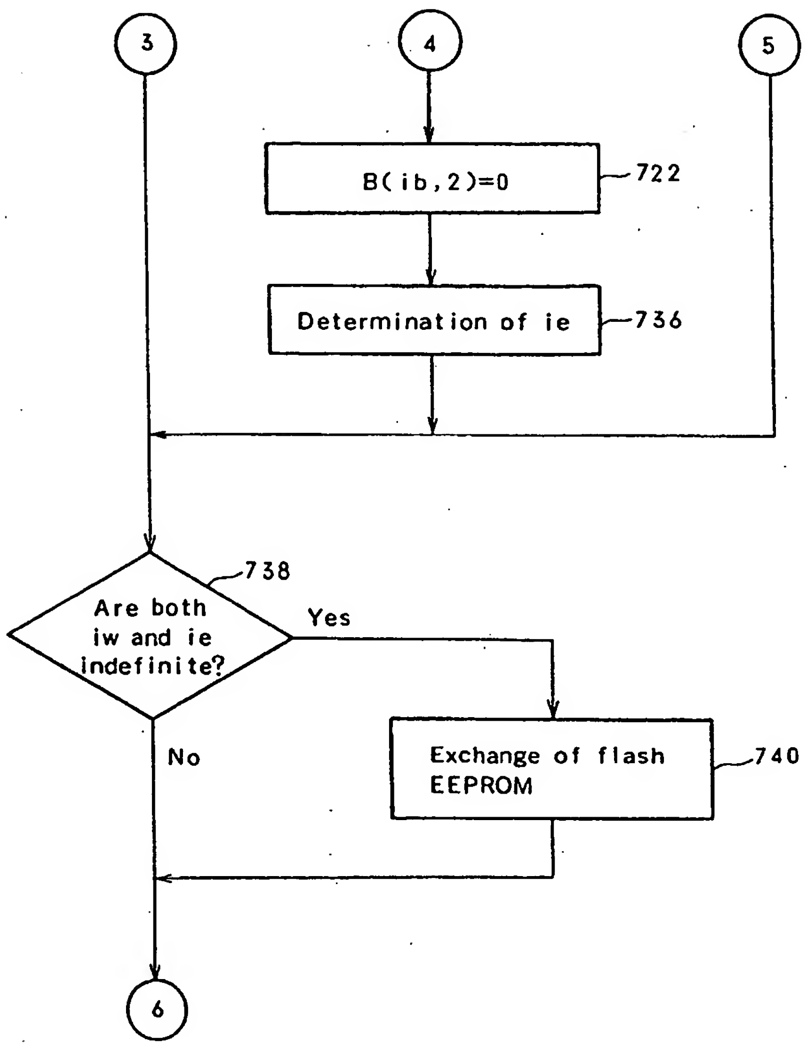

3
4
5
B(ib,2)=0
722
Determination of ie
736
Are both iw and ie indefinite?
738
Yes
No
Exchange of flash EEPROM
740
6

FIG. 8

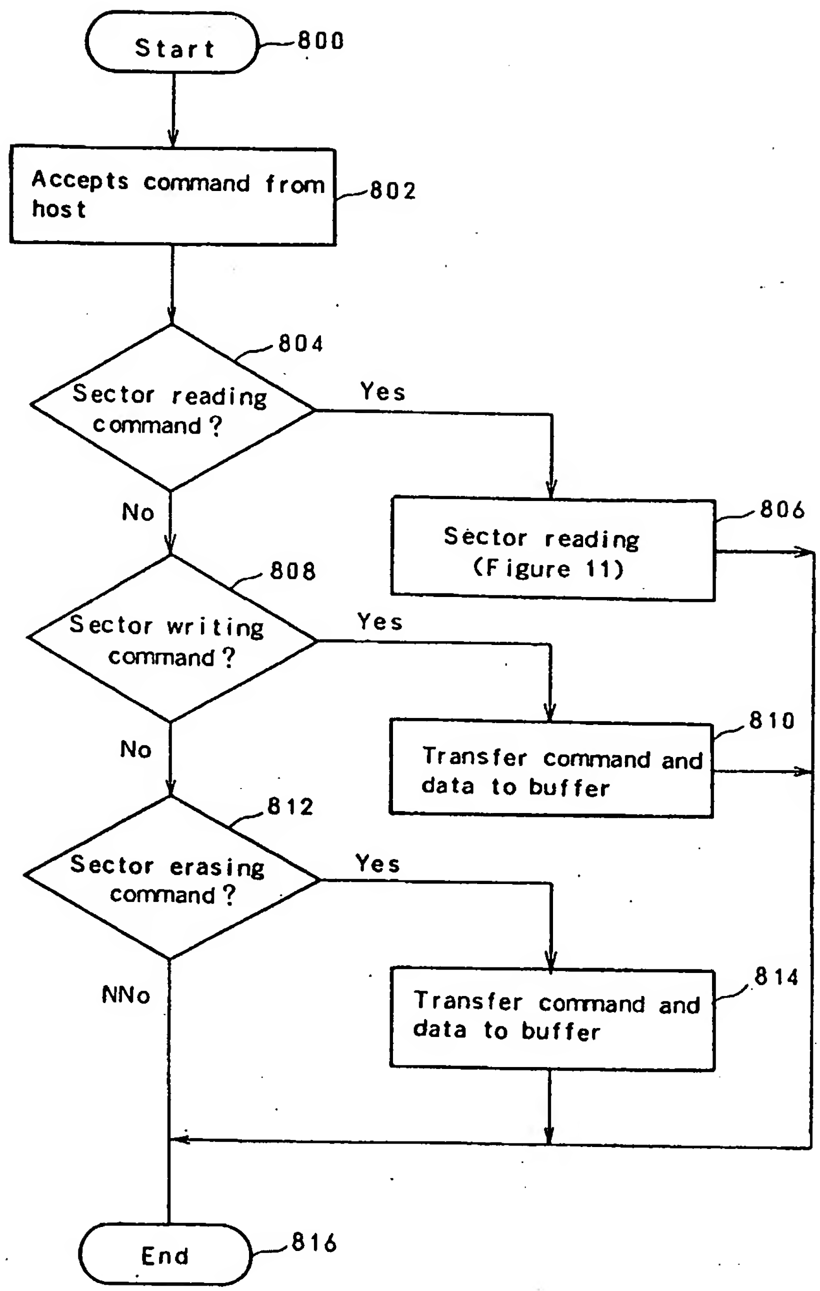

Start
800
Accepts command from host
802
Sector reading command ?
804
Yes
No
Sector reading (Figure 11)
806
Sector writing command ?
808
Yes
No
Transfer command and data to buffer
810
Sector erasing command ?
812
Yes
NNo
Transfer command and data to buffer
814
End
816

FIG.9A

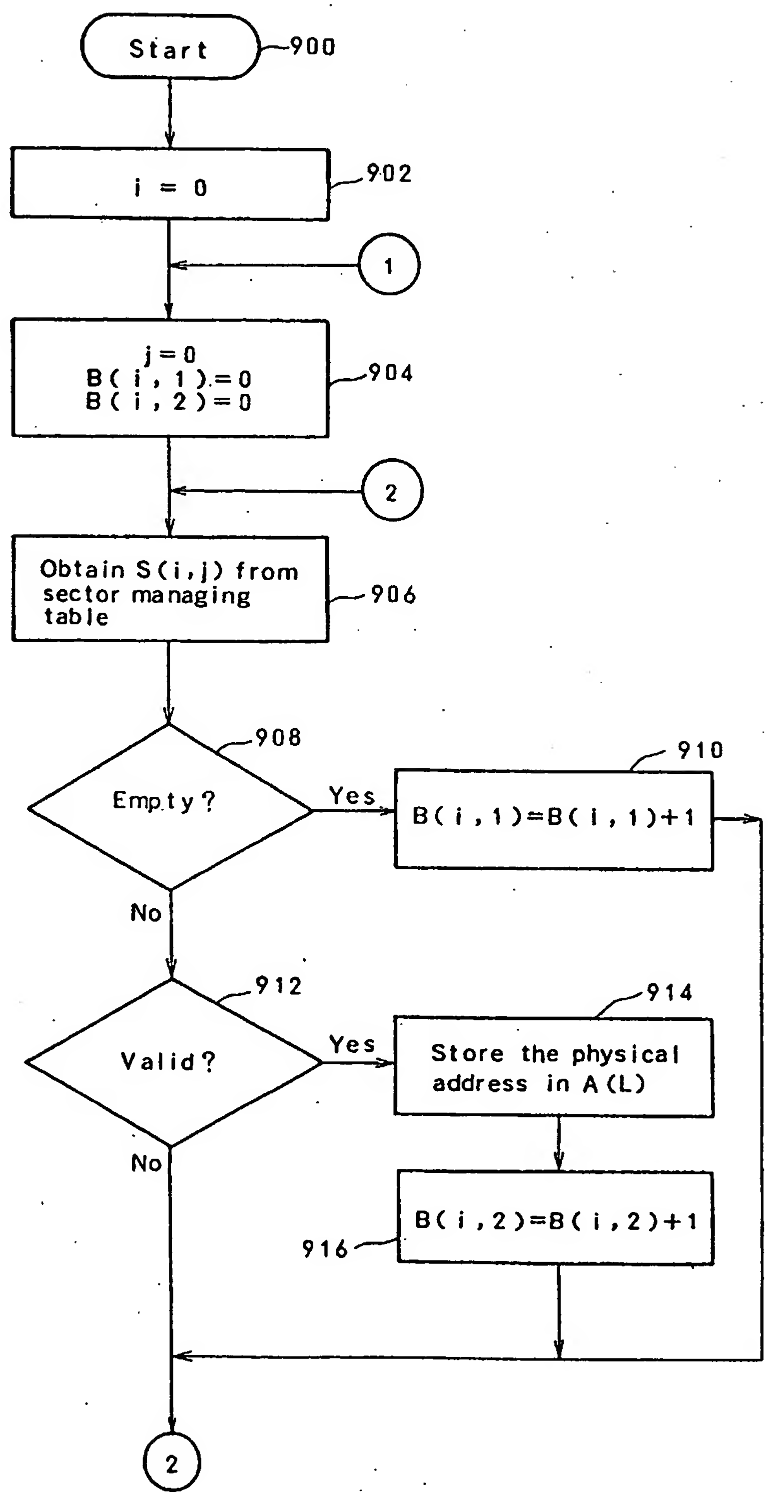

Start
900
i = 0
902
1
j=0
B(i,1)=0
B(i,2)=0
904
2
Obtain S(i,j) from sector managing table
906
908
Empty?
Yes
910
B(i,1)=B(i,1)+1
No
912
Valid?
Yes
914
Store the physical address in A(L)
No
B(i,2)=B(i,2)+1
916
2

FIG.9B

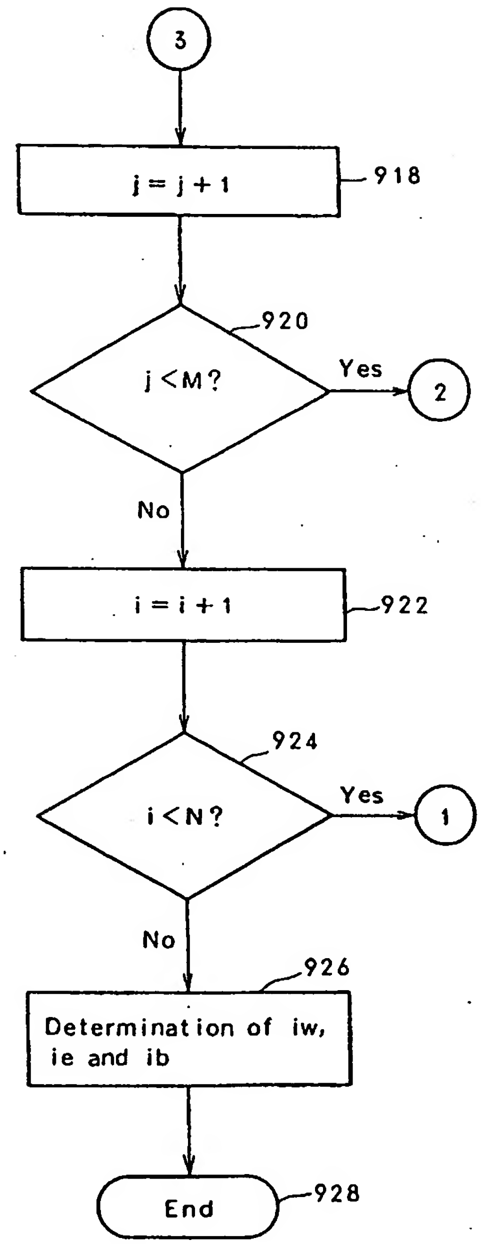

3
j = j + 1
918
920
j < M?
Yes
2
No
i = i + 1
922
924
i < N?
Yes
1
No
926
Determination of iw, ie and ib
End
928

FIG.10

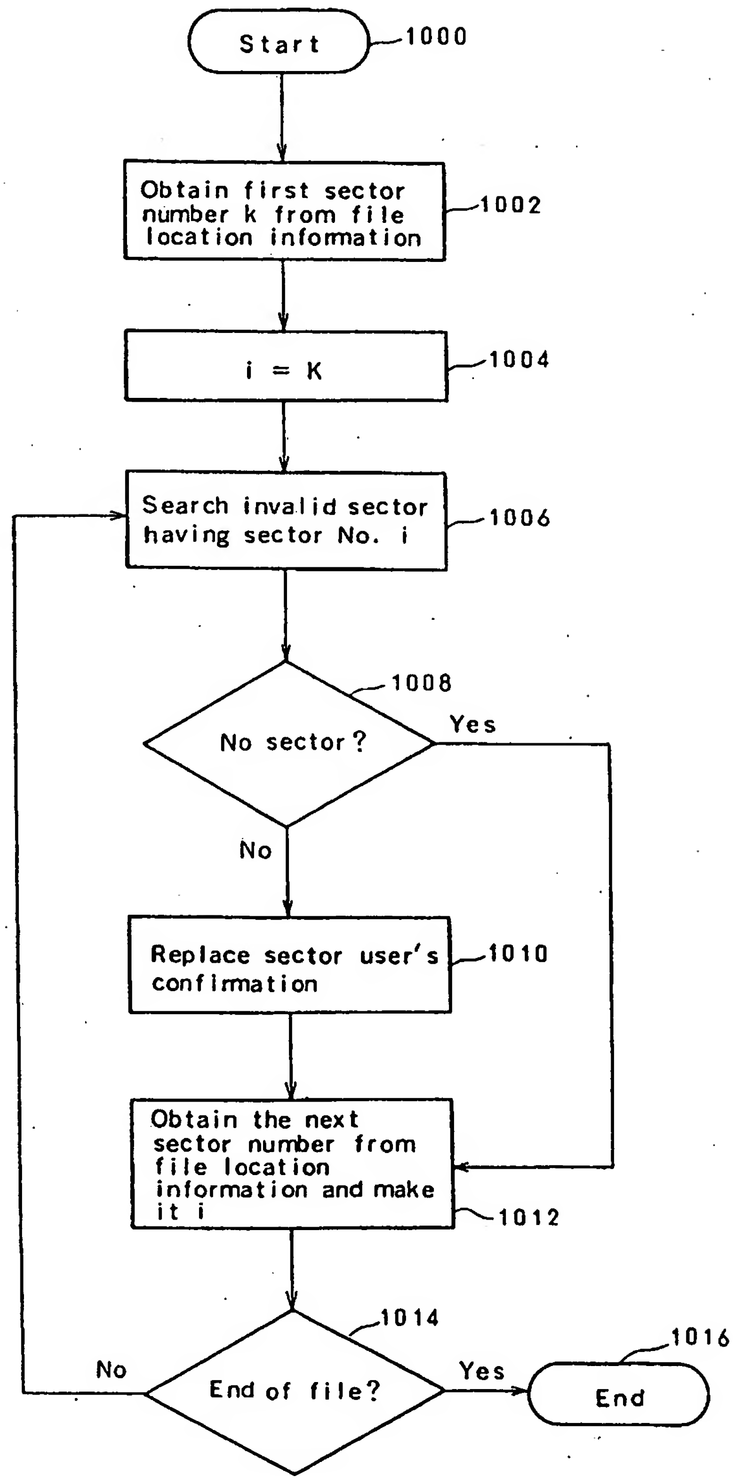

Start
1000
Obtain first sector number k from file location information
1002
i = K
1004
Search invalid sector having sector No. i
1006
1008
No sector?
Yes
No
Replace sector user's confirmation
1010
Obtain the next sector number from file location information and make it i
1012
1014
No
End of file?
Yes
1016
End

FIG.11

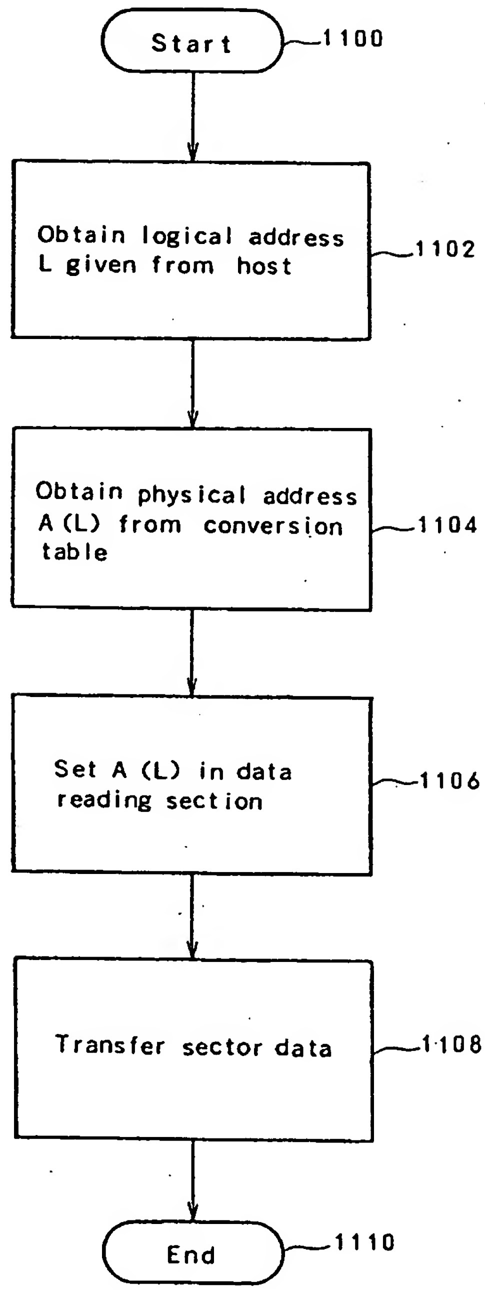

Start
1100
Obtain logical address L given from host
1102
Obtain physical address A (L) from conversion table
1104
Set A (L) in data reading section
1106
Transfer sector data
1108
End
1110

FIG. 12

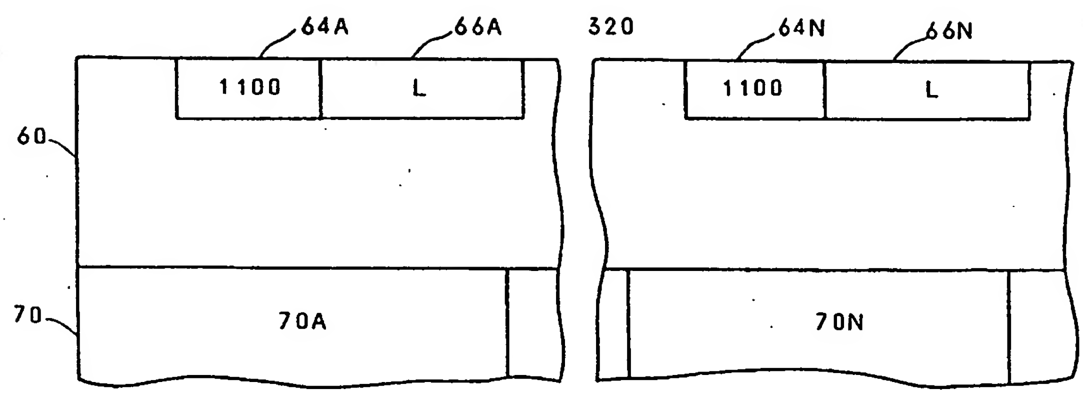


FIG.13A

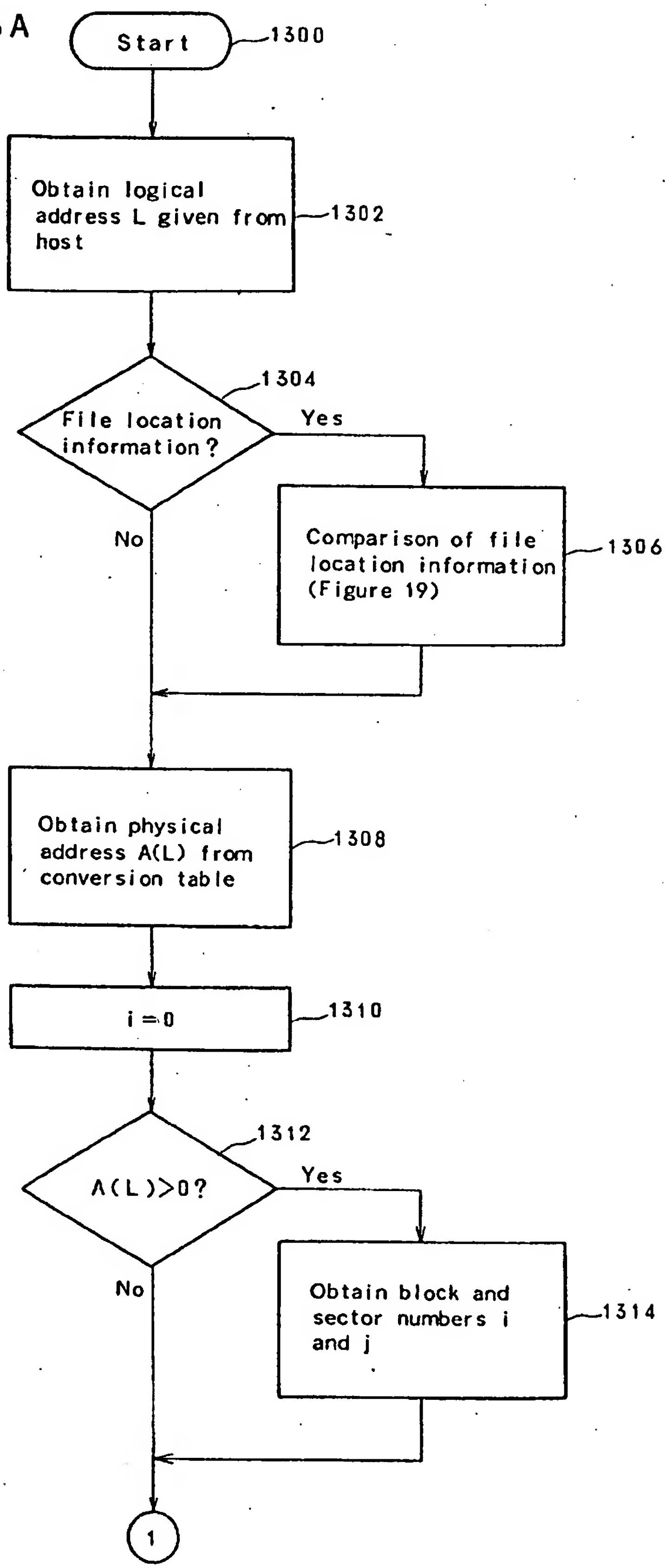

Start
1300
Obtain logical address L given from host
1302
File location information?
1304
Yes
No
Comparison of file location information (Figure 19)
1306
Obtain physical address A(L) from conversion table
1308
i = 0
1310
A(L)>0?
1312
Yes
No
Obtain block and sector numbers i and j
1314
1

FIG.13B

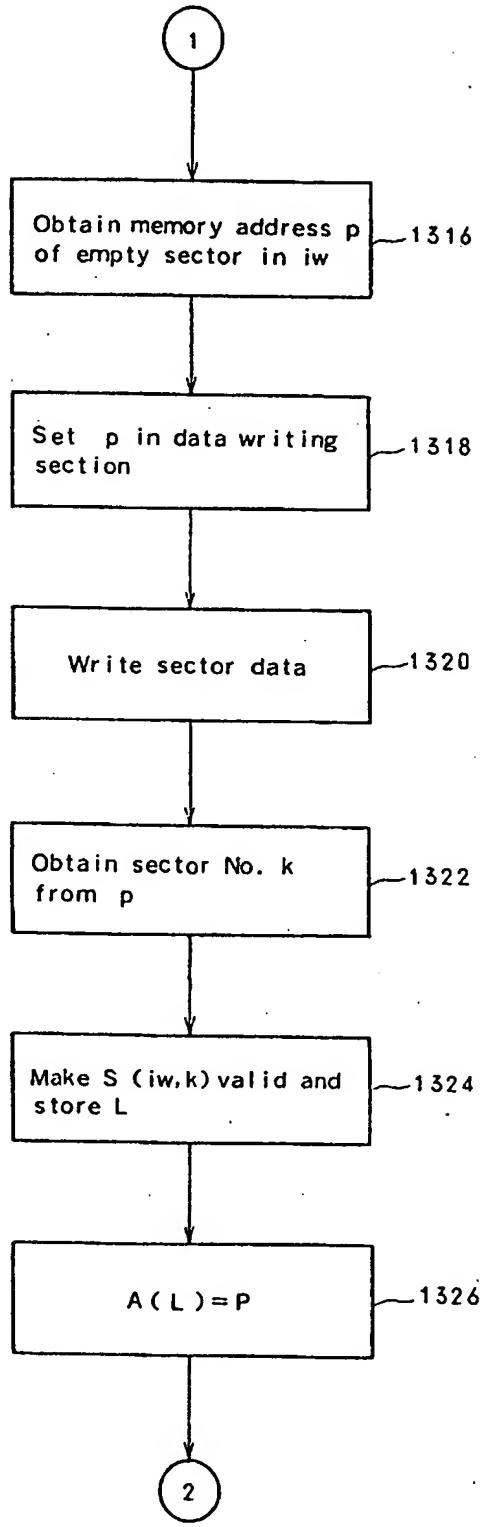

1
Obtain memory address p of empty sector in iw
1316
Set p in data writing section
1318
Write sector data
1320
Obtain sector No. k from p
1322
Make S (iw,k) valid and store L
1324
A(L)=P
1326
2

FIG.13C

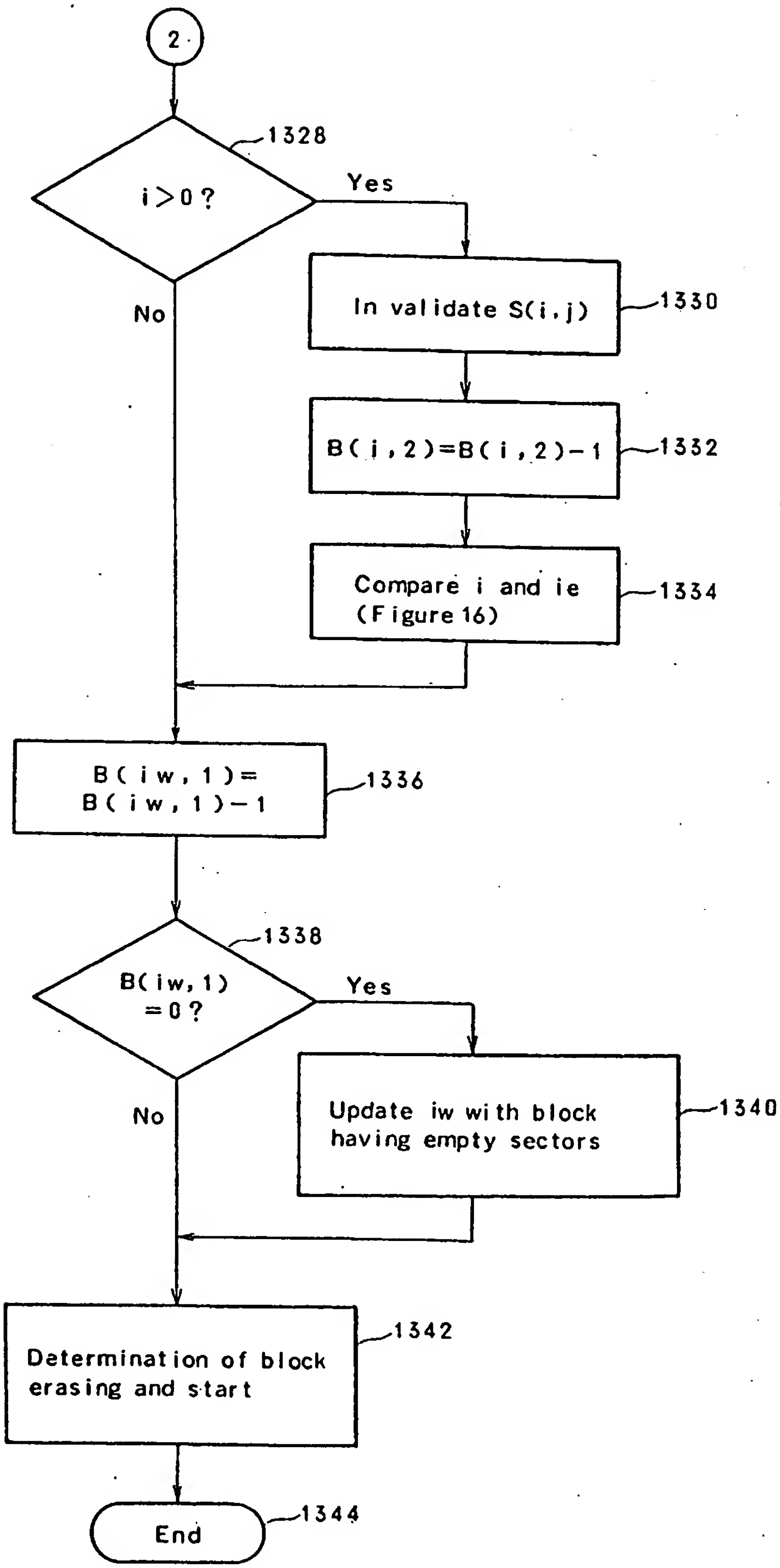

2
i>0?
1328
Yes
No
In validate S(i,j)
1330
B(i,2)=B(i,2)−1
1332
Compare i and ie
(Figure 16)
1334
B(iw,1)=
B(iw,1)−1
1336
B(iw,1)
=0?
1338
Yes
No
Update iw with block
having empty sectors
1340
Determination of block
erasing and start
1342
End
1344

FIG.14

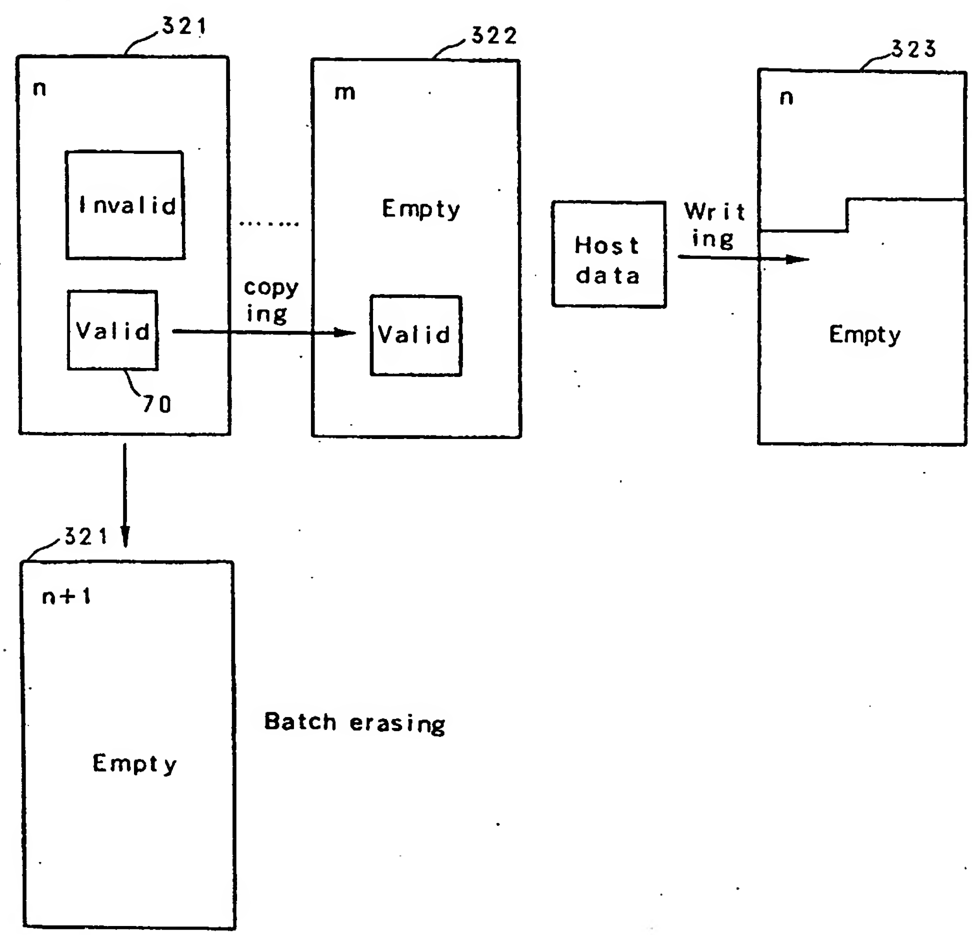

321
n
Invalid
Valid
70
copy ing
322
m
Empty
Valid
Host data
Writ ing
323
n
Empty
321
n+1
Empty
Batch erasing

FIG. 15

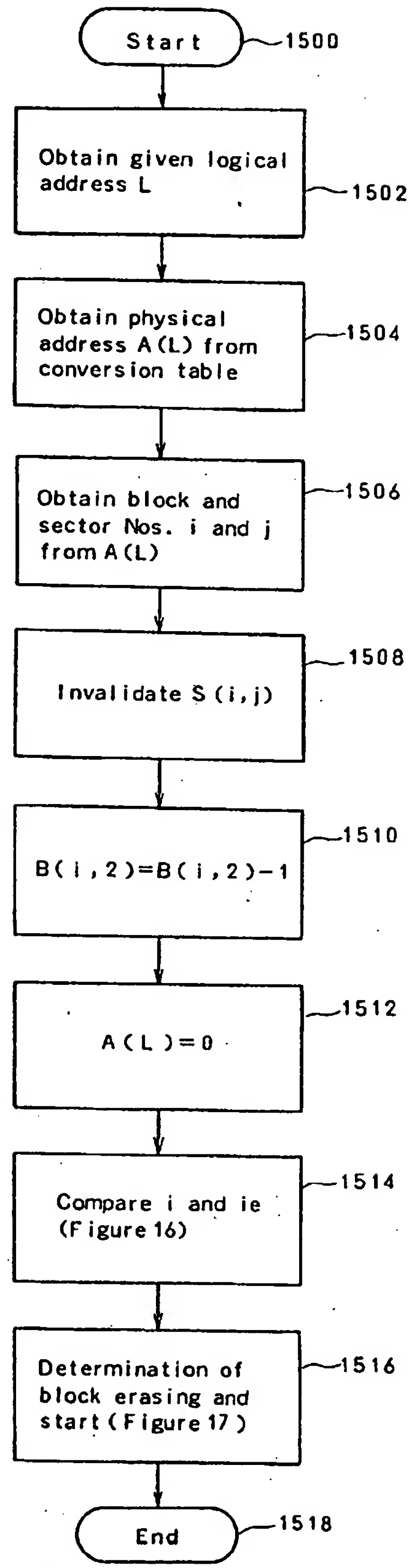

Start
1500
Obtain given logical address L
1502
Obtain physical address A(L) from conversion table
1504
Obtain block and sector Nos. i and j from A(L)
1506
Invalidate S(i,j)
1508
B(i,2)=B(i,2)-1
1510
A(L)=0
1512
Compare i and ie (Figure 16)
1514
Determination of block erasing and start (Figure 17)
1516
End
1518

FIG. 16

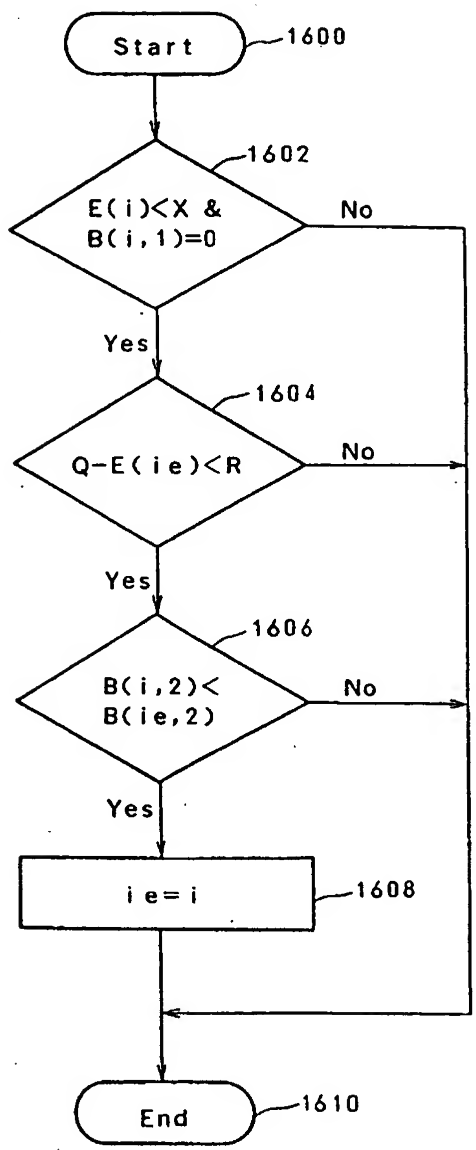

Start
1600
1602
E(i)<X &
B(i , 1)=0
No
Yes
1604
Q−E(i e)<R
No
Yes
1606
B(i , 2)<
B(i e, 2)
No
Yes
i e = i
1608
End
1610

FIG.17

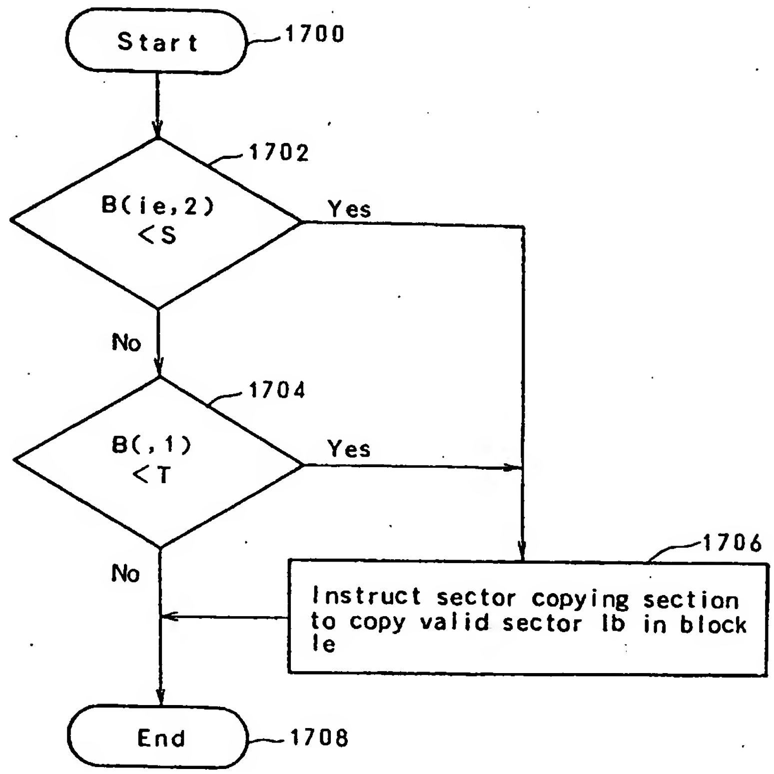

Start
1700
1702
B(ie,2)
<S
Yes
No
1704
B(,1)
<T
Yes
No
1706
Instruct sector copying section to copy valid sector lb in block le
End
1708

FIG.18

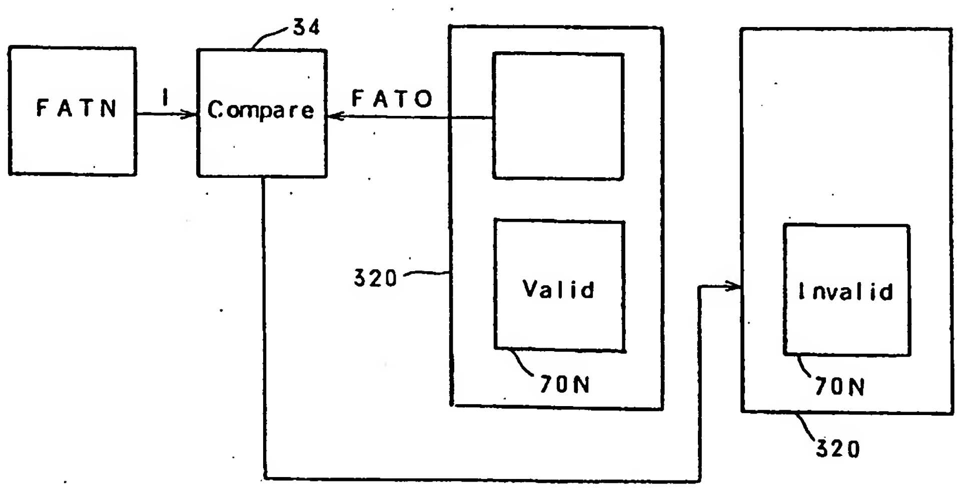

34
FATN
I
Compare
FATO
320
Valid
70N
Invalid
70N
320

FIG. 19

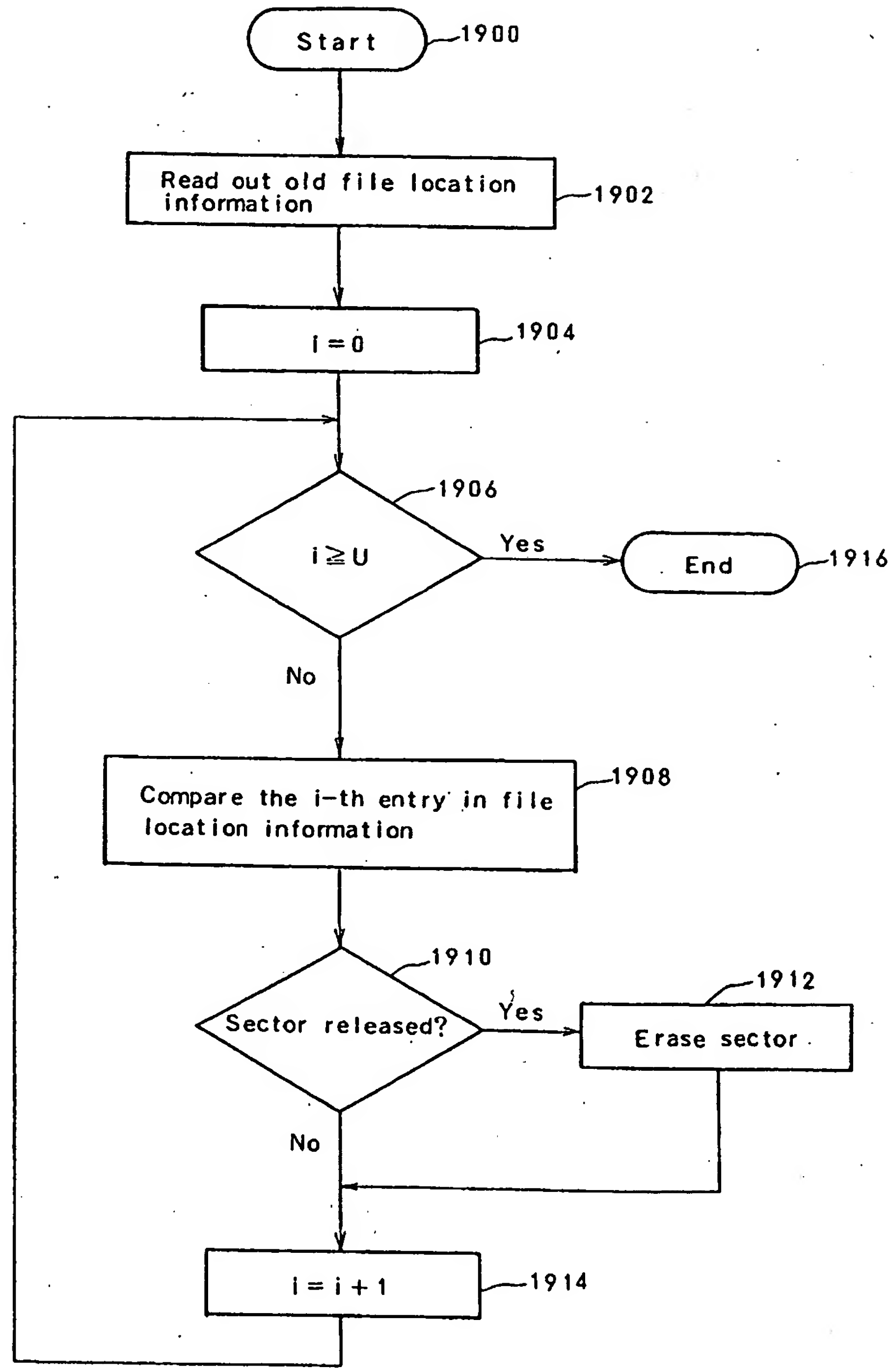

Start
1900
Read out old file location information
1902
i = 0
1904
1906
i ≧ U
Yes
End
1916
No
Compare the i-th entry in file location information
1908
1910
Sector released?
Yes
1912
Erase sector
No
i = i + 1
1914

FIG. 20

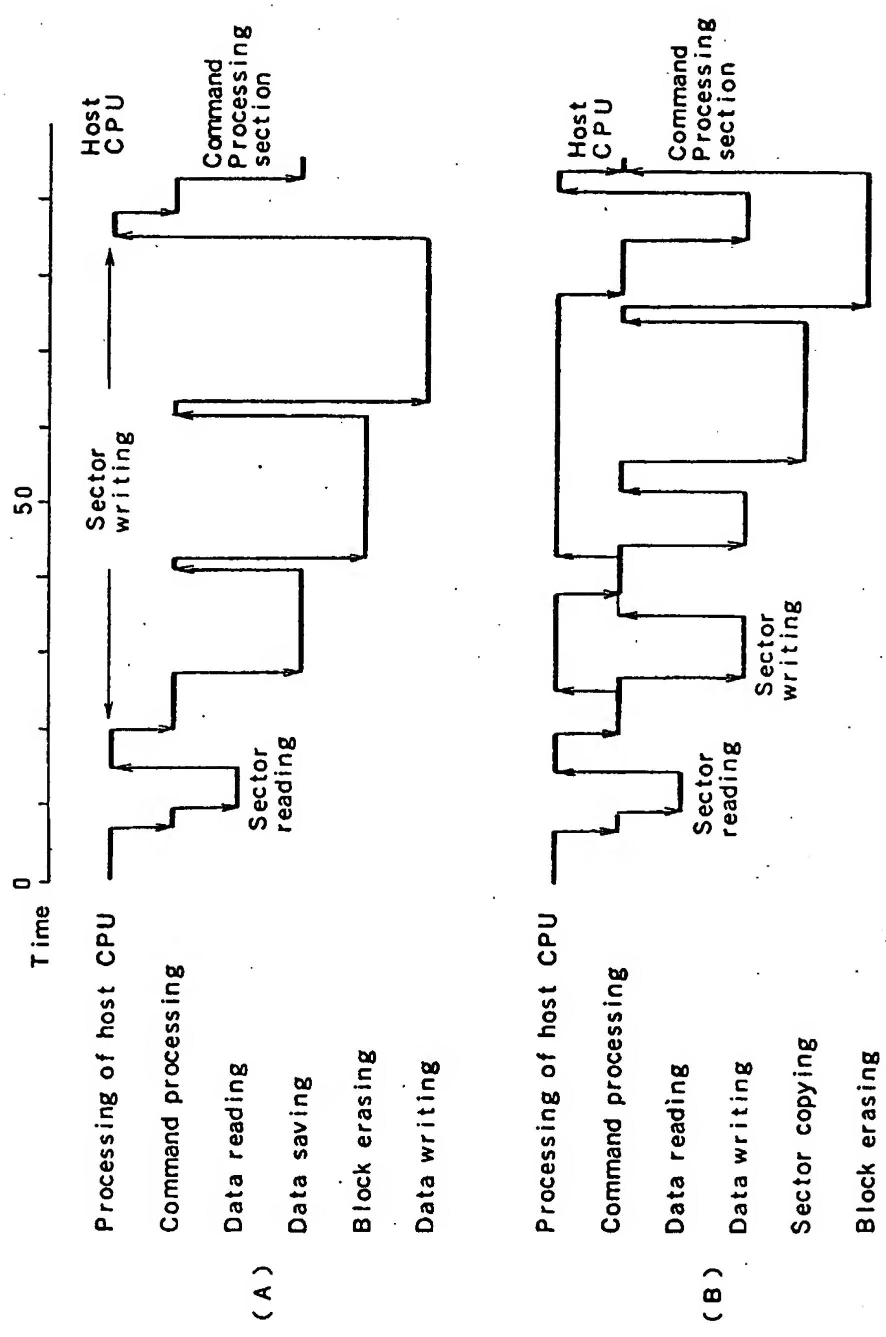

Time
0
50
(A)
Processing of host CPU
Command processing
Data reading
Data saving
Block erasing
Data writing
Sector reading
Sector writing
Host CPU
Command Processing section
(B)
Processing of host CPU
Command processing
Data reading
Data writing
Sector copying
Block erasing
Sector reading
Sector writing
Host CPU
Command Processing section

FIG.21

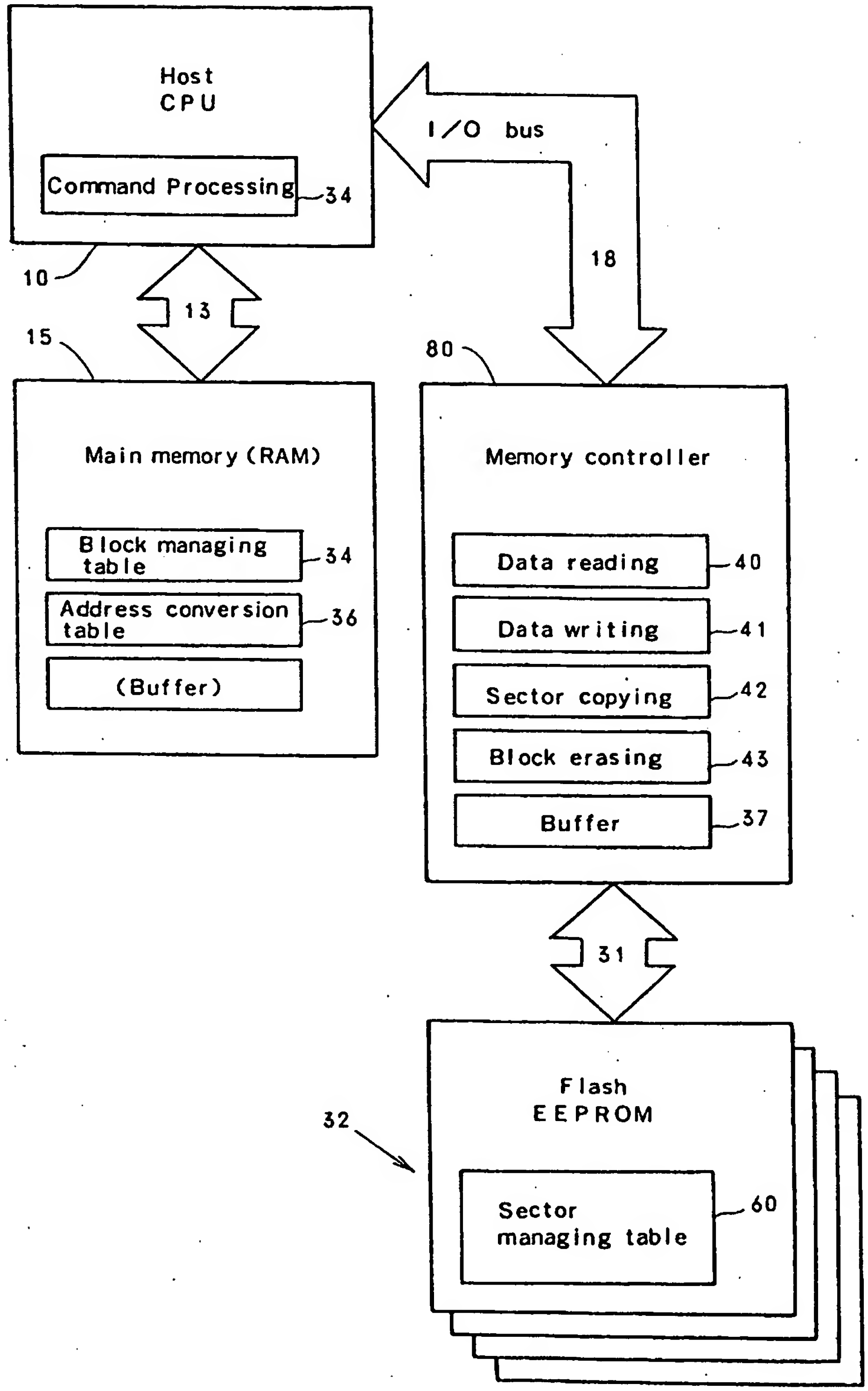

Host
CPU
Command Processing
34
I/O bus
10
13
18
15
80
Main memory (RAM)
Memory controller
Block managing table
34
Address conversion table
36
(Buffer)
Data reading
40
Data writing
41
Sector copying
42
Block erasing
43
Buffer
37
31
Flash
EEPROM
32
Sector managing table
60

FIG. 22

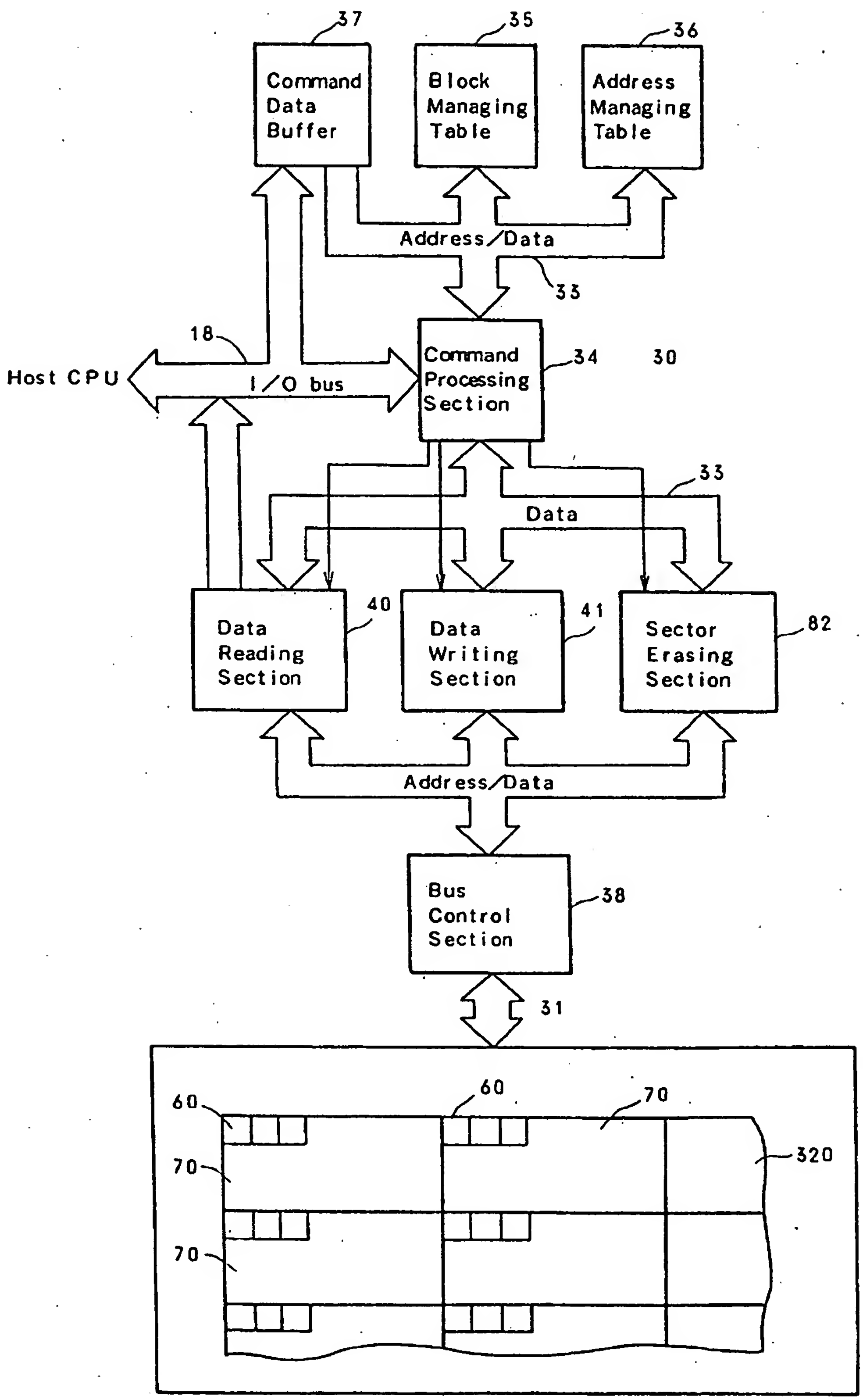

37
35
36
Command Data Buffer
Block Managing Table
Address Managing Table
Address/Data
33
18
Host CPU
I/O bus
Command Processing Section
34
30
33
Data
Data Reading Section
40
Data Writing Section
41
Sector Erasing Section
82
Address/Data
Bus Control Section
38
31
60
60
70
320
70
70

FIG.23

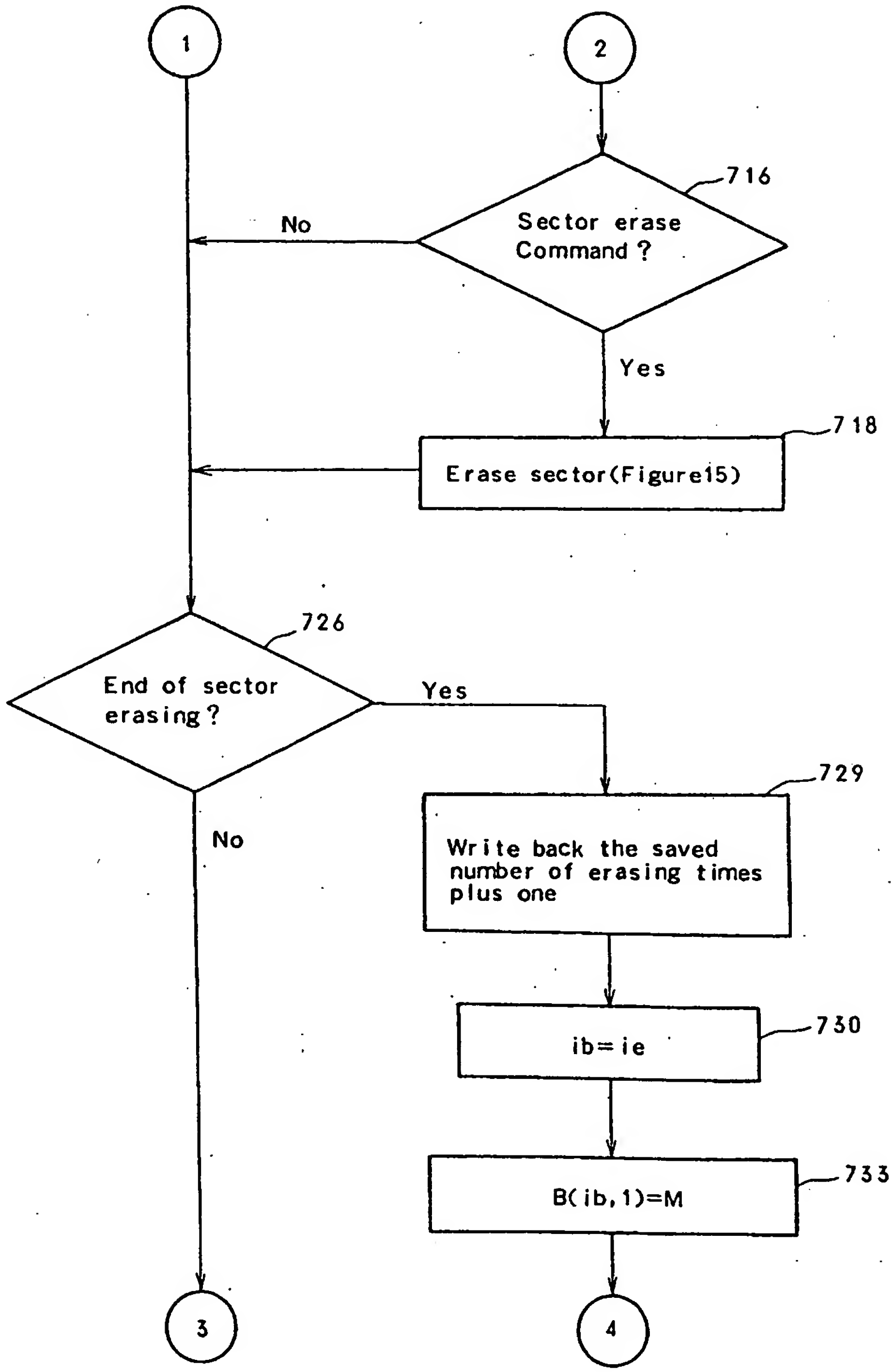

1
2
716
Sector erase Command ?
No
Yes
718
Erase sector(Figure15)
726
End of sector erasing ?
Yes
No
729
Write back the saved number of erasing times plus one
730
ib=ie
733
B(ib,1)=M
3
4

FIG.24

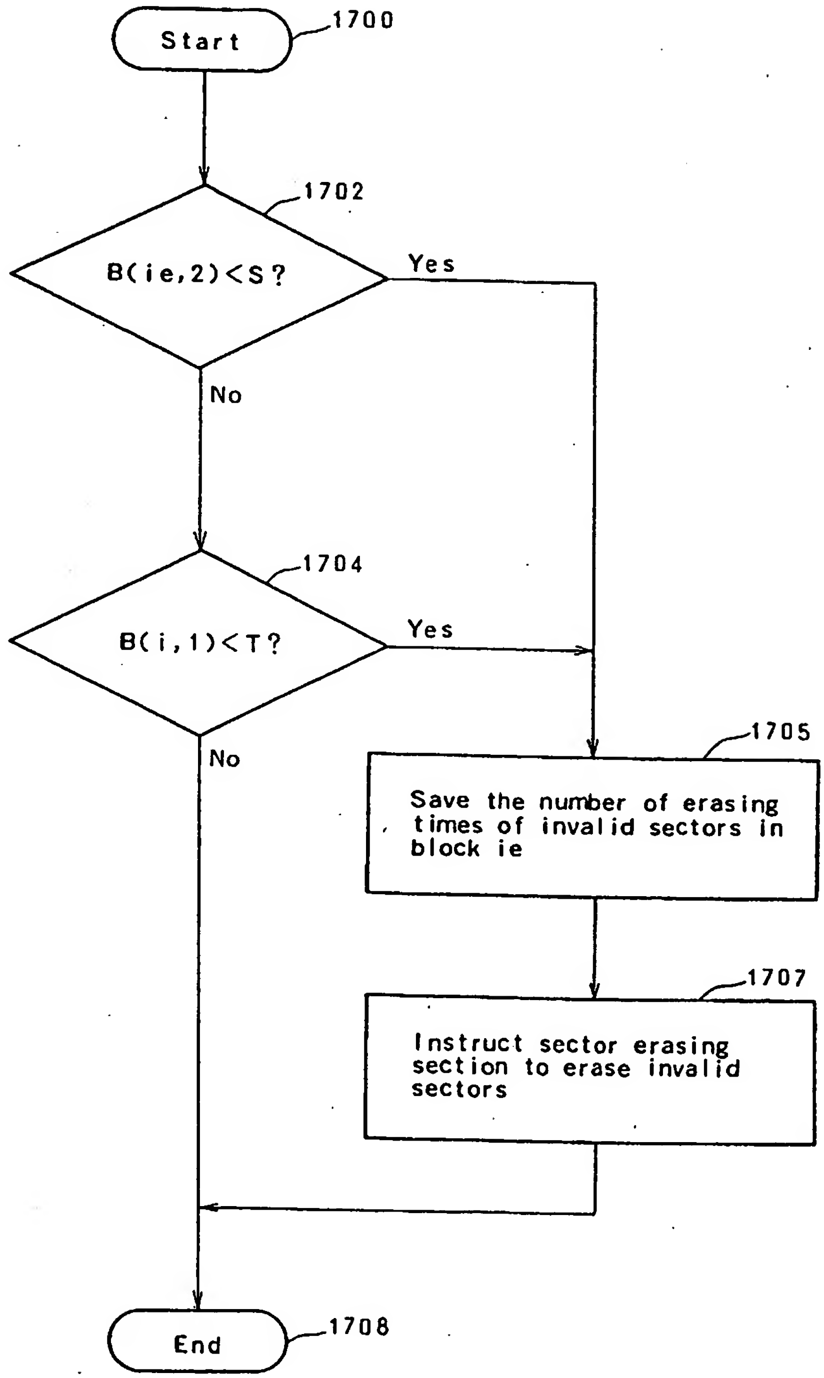
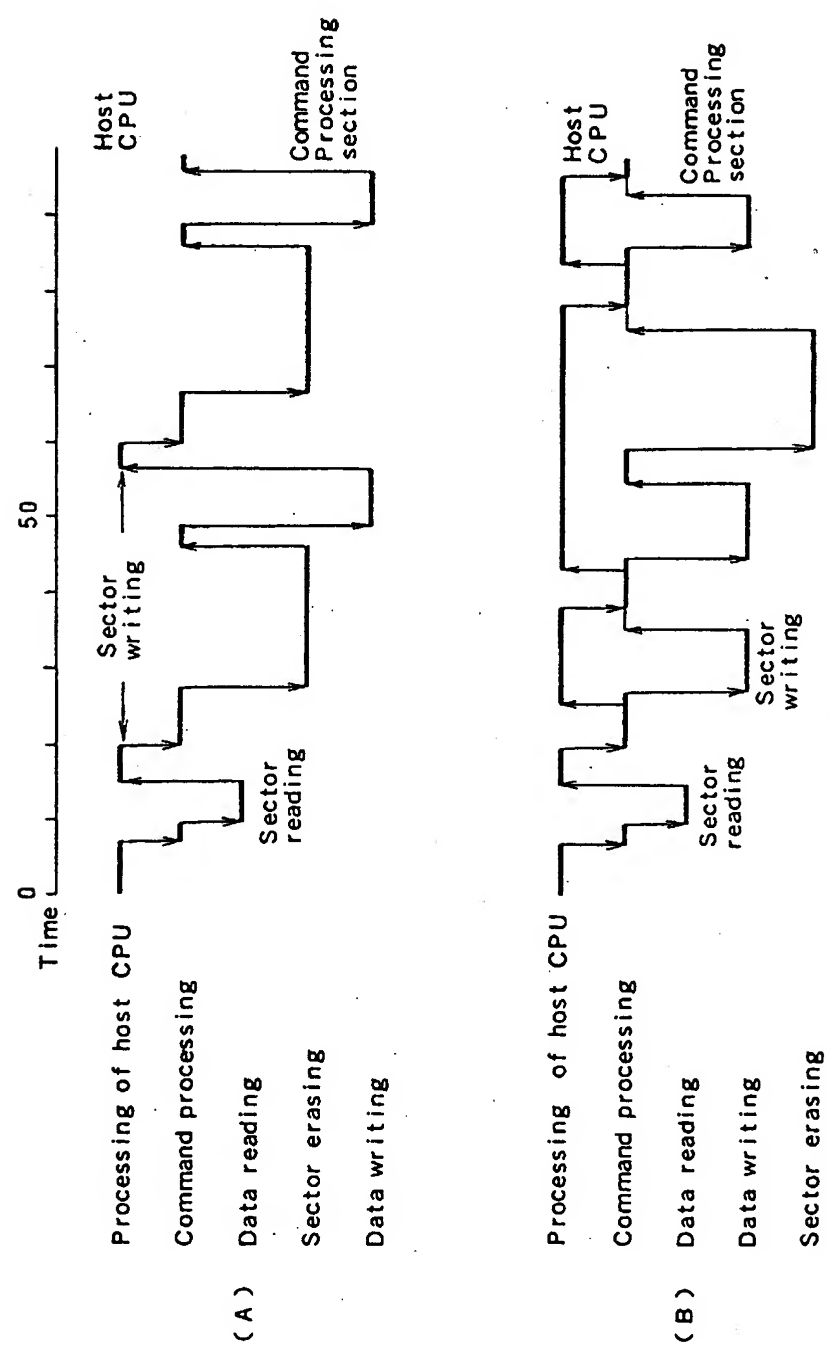

Start
1700
B(ie,2)<S?
1702
Yes
No
B(i,1)<T?
1704
Yes
No
Save the number of erasing times of invalid sectors in block ie
1705
Instruct sector erasing section to erase invalid sectors
1707
End
1708

FIG. 25


Time
0
50
(A)
Processing of host CPU
Command processing
Data reading
Sector erasing
Data writing
Sector reading
Sector writing
Host CPU
Command Processing section
(B)
Processing of host CPU
Command processing
Data reading
Data writing
Sector erasing
Sector reading
Sector writing
Host CPU
Command Processing section